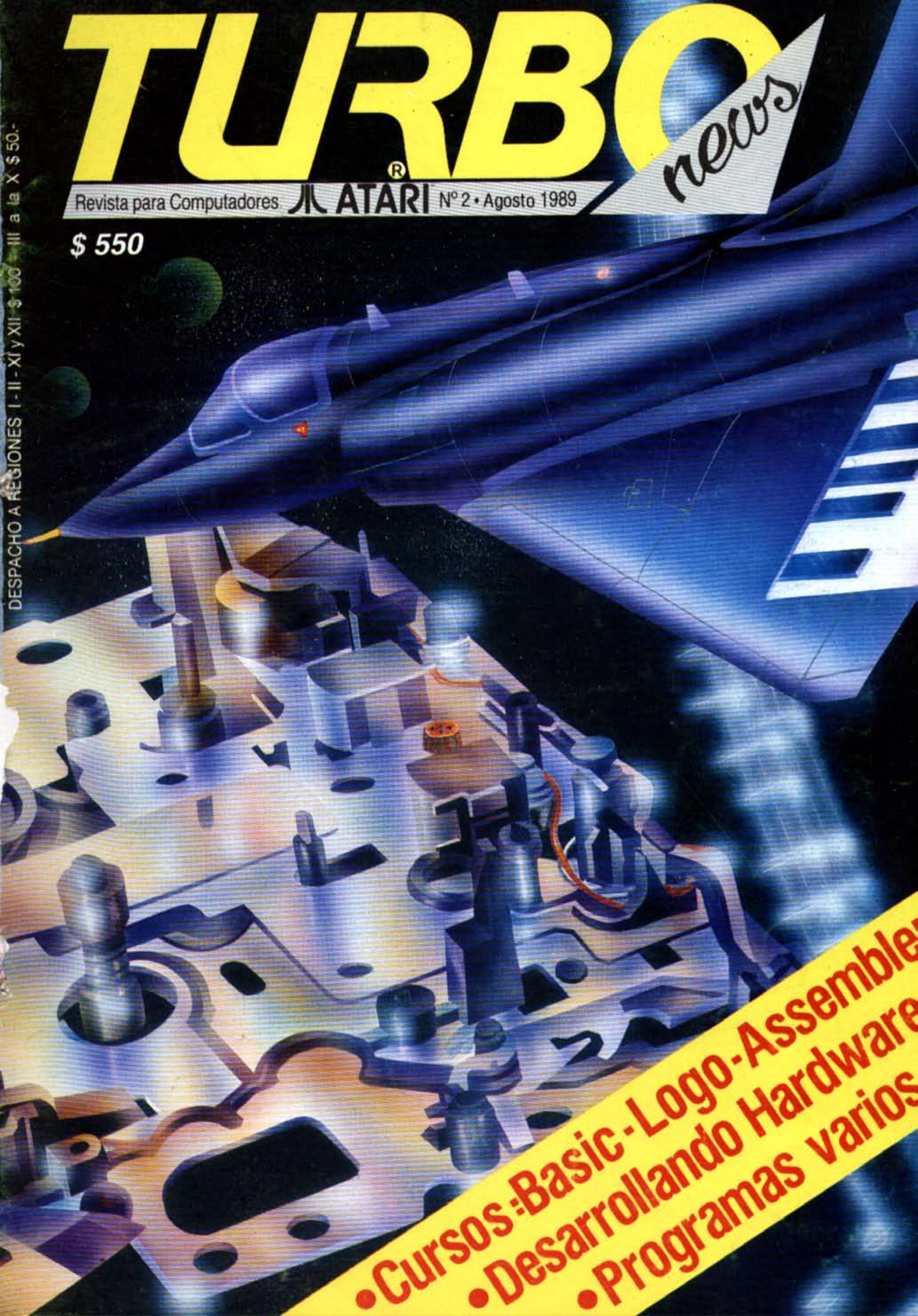
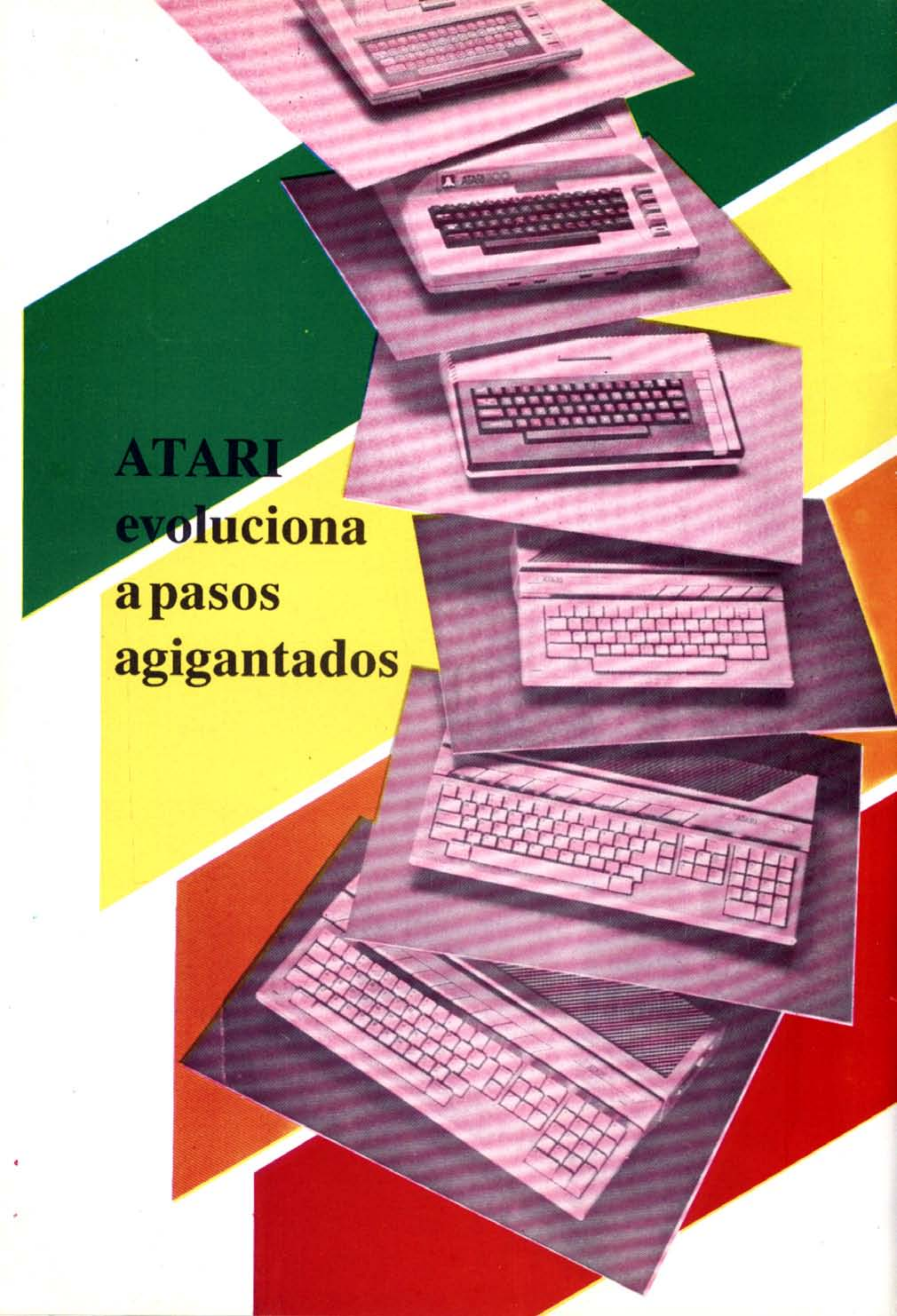

TURBO news
Revista para Computadores ATARI Nº 2 • Agosto 1989
$ 550
DESPACHO A REGIONES I - II - XI y XII $ 100 - III a la X $ 50.-
•Cursos:Basic-Logo-Assembler
•Desarrollando Hardware
•Programas varios



ATARI
evoluciona
a pasos
agigantados





Circulación mensual. Nacional e Internacional
Destinada a los usuarios de computadores ATARI®como material didáctico de PROGRAMACION TURBO news®
es una publicación de EDITORA TURBO LTDA.
Domicilio Av. Holanda 2456 - Teléfono: 2238063
SANTIAGO CHILE.

GERENTE: Arturo Valdivieso Martínez. **EDITOR:** Editora Turbo Limitada. **PRODUCCION:** Pedro P. Caraball Alvarez - Marcelo A. Waldbaum Olszevicki - Mauro Pieressa Schachtel, Programadores y Diseñadores de Computación **COLABORADORES:** Mariana Pizarro - Francisco Azagra. **PERIODISTA:** Rodrigo Manríquez **ARTE/DISEÑO:** Odalí Guerrero López - Diseñadora Gráfica. **FOTOGRAFIAS:** Isabel Bellalta.

DEPARTAMENTO DE VENTAS Y PUBLICIDAD: Arturo Valdivieso M. Hernán Vittini G. Agradecemos la colaboración de: Coelsa S.A. Centro Atari (Augusto Leguía Sur 75). ATARI, es marca registrada de Atari Corporation. TURBO, es marca registrada de EDITORA TURBO LIMITADA (Reg. Marc. N°342428 9-05-89). Impresa en los talleres de Impresos Nova Ltda. quien actúa como Impresora.

El mundo cambia vertiginosamente. Lo que se hacía ayer, hoy se puede hacer antes y mejor. Mañana se podrá obtener el mismo resultado por un camino aún más corto.

Los avances van en todos los sentidos, pero el problema es saber manejarlos. Ante ello nuestra revista tiene un rol que asumir. Queremos contribuir estableciendo un fructífero intercambio de ideas entre todos los aficionados a la computación.

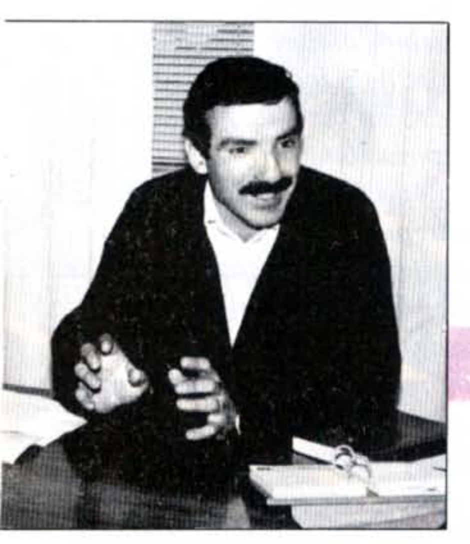
Aprenderemos juntos y compartiremos novedades que nos permitirán ir conociendo cada vez mejor el mundo de la informática. Si nuestro objetivo se cumple, pondremos un grano de arena para que la persona controle a la máquina y no a la inversa.

Con este fin en el número de agosto les presentamos cursos y actividades para desarrollar habilidades manuales y saber más sobre los poderosos Atari.

La nueva tecnología tiene un gran aporte que entregar al progreso sin embargo, para que pueda ser aprovechada, debe conocerse.

LOGO

EL LENGUAJE DE LA ENSEÑANZA



"Con esto seguro que me van a causar un infarto", pensó el profesor. "Y con lo caros que son esos aparatos, capaz que los echen a perder".

Al viejo maestro no le agradó la idea de que sus alumnos, a los que consideraba demasiado inquietos manejara complejos equipos de computación. Ya se imaginaba desenredando una nube de cables o a los estudiantes del octavo básico corriendo entre los costosos monitores.

Sin embargo, nada de eso ocurrió. Todo lo contrario. Cuando entró a la sala vió a sus educandos transformados. Zelada, que siempre le tiraba el pelo a las niñas, permanecía silenciosamente boquiabierto. Moreno y Aguirre conversaban suavemente frente a una pantalla.

El éxito de la computación surge de la atracción que la máquina, como tal, produce.

Todos estaban sentados con una calma expectante. En esos instantes la opinión del docente respecto a las máquinas cambió.

Así, hace ya algunos años, se registraron los primeros encuentros entre las comunidades educativas y la informática.

El resultado ya se nota con la difusión masiva de la nueva tecnología. Su éxito parte del impacto social que la idea "computador" causa. Esta es muy poderosa, pues crea una motivación por conocerla en amplios grupos de la población. Es una de las pocas cosas en que todos -jóvenes o viejos, pobres o ricos, derechistas o izquierdistas- están de acuerdo.

Según el psicopedagogo de la Universidad de Concepción, Pablo Saba Cadenas, "hace pocos años, cuando se inventó el motor eléctrico, era una gran revolución. Ahora lo estamos usando en todas partes y pasa como muy transparente.

La informática es así. Es, después de la escritura, un vuelco total, otra revolución".

Exploración y Descubrimiento

Tradicionalmente se enseña recurriendo a la memorización de conceptos. Para Pablo Saba "cuando no parte del interés del mismo sujeto, el conocimiento puede pasar inadvertido o ser impuesto por el medio". La segunda alternativa implica aprender de acuerdo a estructuras formales que no surgen del carácter de cada persona, sino del promedio de un grupo que es un curso. El individuo queda limitado en sus intereses por un determinado currículo. Así se favorece a quienes tienen rasgos destacados en este esquema.

El computador es un avance en los métodos de educación pues la atracción que produce permite un conocimiento propio, adoptado internamente. Lo que así se aprende no se olvida.

El psicopedagogo dice que ha elegido el lenguaje "Logo" para realizar un proyecto de enseñanza, argumentando que el "Logo" "está hecho por educadores".

Al usar los primeros comandos el niño logra resultados y su interés es retroalimentado. La internalización del conocimiento se obtiene cuando el niño explora un área determinada y descubre algo. Eso queda grabado en la memoria.

Trabajando en un computador se previene una descalificación al escolar. La máquina no puede lanzarle una mirada o cualquier expresión que lo descalifique emotivamente. Sólo da respuesta a los códigos que ingresan.

Así llega el momento en que el estudiante puede hacer transferencias. Esto implica que puede cambiar la estructura conceptual de una situación a otra. "Va a depender", declara Saba, "del mediador, del profesor que esté allí guiando el aprendizaje, manejando algunas variables para que el niño comience a desarrollar algún tipo de estrategia congnitiva. Si lo dejas solo seguramente va a aprender muy poco o va a aprender de su cercana experiencia. Ahora si le enriqueces su ambiente y lo llenas de experiencias en las cuales pueda hacer transferencias, va a ser un niño con un buen repertorio. Va a desarrollar mejores habilidades para desenvolverse en la vida".

El psicopedagogo enfatiza que "Logo" "es como una extensión de la mente, es como un espejo donde los educandos ven sus avances y sus proyectos. Entre comillas, ven el proceso de como aprenden a aprender. Digo entre comillas porque no está comprobado científicamente, sino basado en la observación y la experiencia".

La tecnología moderna no garantiza que no haya riesgos. Uno muy importante es la desmotivación, que es más fácil de producir en los niños más chicos. Por eso Saba recomienda que los cursos de "Logo" se inicien en tercero básico y continúen hasta octavo. La misión del profesor en ese marco es guiar al estudiante para que tenga motivaciones constantes a seguir aprendiendo.

El Computador, de Peñalolén a las Rejas

Este método se aplicó en la Escuela 187, de Peñalolén, al igual que en otras tres de esa comuna. En total 2453 alumnos.

El resultado, en opinión de Saba, se notó. El profesional señaló que un estudio con niños de un barrio pobre de Nueva York, a los que se les enseñó computación "mostró en ellos un cambio positivo y una baja en la parte agresiva. Nosotros también recibimos nuestra experiencia, aunque no está aún cuantificada. Una profesora citó a los padres a una reunión y los llevó a la sala de computación para trabajar con ellos. Quedaron muy entusiasmados y le pidieron otras clases". Los hijos también reaccionaron. La asistencia a las jornadas en que hay talleres aumentó.

"Ayer", añade Saba, "me pasó por segunda vez. Un niño que estaba faltando, por que tenía bronquitis y una otitis bien fuerte, fue a una clase de computación. Le tocaba en la última hora y la mamá lo llevó para que después volviera a casa. Eso mostró mucha motivación".



Entre los origenes de esta actividad está una reunión realizada en 1987. Allí el psicopedagogo emprendió iniciativas en la informática junto a un equipo multidisciplinario. El grupo fue integrado por Gustavo Jiménez Lagos, psicólogo y master en computación; María Cristina Escobar, ingeniero en computación; Miguel Vera, profesor de física; y Eduardo Arancibia, programador en computación.

Surgió así un proyecto cuya metodología integra aspectos de "exploración y descubrimiento" con la guía de un mediador. Se diseñaron instrumentos para medir las siguientes variables: creatividad, exploración, análisis, planificación y corrección de errores. Estas se aplicaron en 15 sesiones para cursos de tercero a séptimo básico. El trabajo incluyó a cerca de 1500 alumnos de cinco colegios. El objetivo era la adquisición de habilidades científicas.

El año pasado la Subsecretaría de Desarrollo Regional Ministerial hizo propuestas públicas para equipar laboratorios de computación en corporaciones municipales de educación en todo Santiago. Coelsa, la distribuidora de productos Atari, se adjudicó cerca del 50 por ciento. Esta empresa asumió los estudios realizados por el equipo de Saba. Con ellos se elaboró un plan de cuyos objetivos fueron definidos como: capacitar en "Logo" a profesores, aumentar la asistencia a clases de los niños y acrecentar su autoestima. Además se iniciaron programas de reforzamiento.

La actividad se repartió en diez municipalidades. En diciembre culminó la formación de varios grupos de maestros, cerca de cien en total, aproximadamente una decena por corporación.

Saba concluyó diciendo: "con la experiencia que estos profesores obtengan en el primer año del proyecto ya se puede trabajar con alumnos más pequeños".

En el artículo anterior, hemos visto como "piensa" un computador. Para ello nos hemos valido de una técnica gráfica denominada "Diagramas de Flujo". En este número conoceremos las primeras instrucciones del lenguaje Basic y veremos cómo unirlas para formar programas sencillos. Aplicaremos en la confección de dichos programas, la técnica de graficación anteriormente mencionada.

Para comenzar, recordaremos brevemente lo visto en el número anterior. Para la confección de todos los gráficos, estamos utilizando dos símbolos, rectángulo, que indica una orden y rombo, que indica decisión, unidos por líneas horizontales y verticales que indican el "camino" que lleva la ejecución del programa. A esto hay que agregarle un símbolo indicando el comienzo y otro, que pueden ser varios o ninguno, de fin. Para aquellos que se incorporen en esta etapa, no deben preocuparse si al principio les resulte complicado, en la práctica verán la sencillez de esta técnica.

Una vez recordado esto, podemos empezar a ver las primeras instrucciones del Basic.

Comenzaremos definiendo lo que es una variable y lo que es una constante. Constante son aquellas cosas que no cambian de valor con el tiempo, por ejemplo el número 4 es una constante que siempre vale 4.

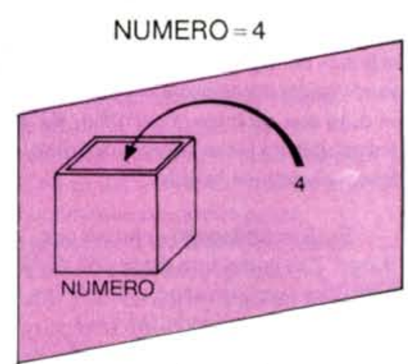
Una variable podemos asemejarla a una caja vacía, cuyo contenido VARIA en el tiempo. A dicha caja hay que ponerle un nombre y esa va a ser la identificación de dicha variable. Por ejemplo, a la variable NUMERO se le puede asignar el valor 1 posteriormente el 2 y etc. pero la caja siempre se va a llamar NUMERO. El nombre de la caja puede ser cualquiera aunque se recomienda que guarde una relación con el contenido. Por ejemplo una variable donde se vaya a almacenar la edad de una persona convendría llamarla EDAD. Atari, al igual que todos los computadores, tiene ciertas limitaciones en cuanto a los nombres posibles, como por ejemplo el que debe comenzar obligatoriamente con una letra o ser escrito en mayúsculas. Otro ejemplo de nombres de variable serían: MAYOR, MENOR, NOMBRE, DOMICILIO, CAJA, A, B, etc.

Las constantes, en cambio, no tienen ninguna limitación, pueden ser mayúsculas, minúsculas, símbolos, números, etc.

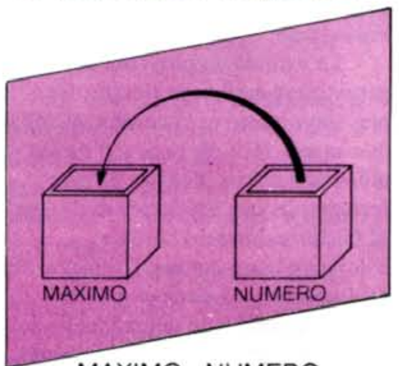
Existe una clasificación dentro de las variables y las constantes. Estas pueden ser numéricas o alfanuméricas, según estén compuestas por números únicamente o letras y números mezcladas. Por ejemplo "18 años", es una constante alfanumérica y 18 es una constante numérica (Las constantes numéricas no necesitan las comillas).

A continuación estamos en condiciones de ver las primeras instrucciones de basic. Para ejecutar una instrucción cualquiera, basta con ingresarla y oprimir la tecla RETURN. La primera es justamente la que permite ir dándole distintos valores a las variables y se denomina asignación. Un ejemplo de la misma sería:

NUMERO = 4



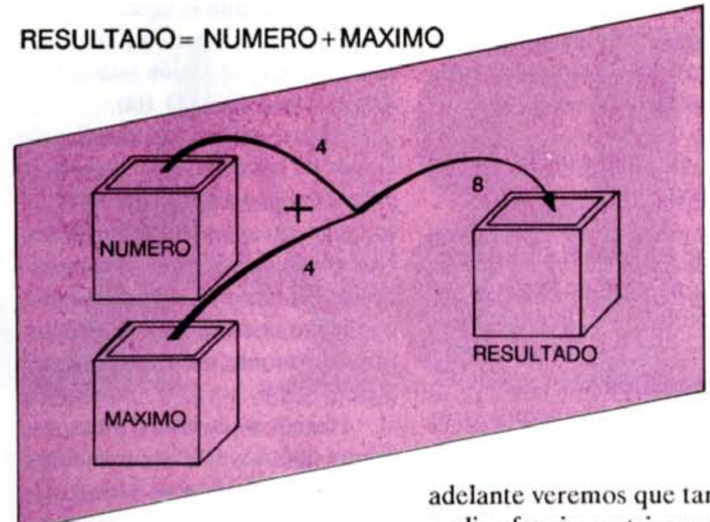
Con esta instrucción la variable NUMERO toma el valor 4. Podemos hacer lo siguiente:



MAXIMO = NUMERO

En este caso el computador toma el valor contenido en la variable número y lo traspasa a la variable máximo. Esto permite también realizar las primeras operaciones matemáticas. Si hacemos:

BA



estamos asignando a la variable RESULTADO el contenido de NUMERO sumado al de MAXIMO. Después de ejecutar este grupo de instrucciones la variable RESULTADO toma el valor 8. Lo mismo puede hacerse para la resta, multiplicación, división y potencia como se ven en los siguientes ejemplos:

A=B-C en A queda el resultado de restarle a B el valor de C

A=B*C se utiliza el símbolo asterisco para representar la multiplicación quedando en A el resultado de multiplicar B*C

A=B/C queda en A el resultado de dividir B por C

A=B^3 queda en A el resultado de elevar B al cubo

Ya ves la primera aplicación que puedes darle a tu computador, funciona exactamente igual a una calculadora. Aquí hemos visto las funciones más sencillas, más adelante veremos que también realiza funciones trigonométricas, de raíz cuadrada, generación de números al azar, redondeo de cifras, etc.. Es importante destacar que no se puede asignar una variable numérica a una alfanumérica ni viceversa. Las variables alfanuméricas se reconocen porque deben finalizar obligatoriamente con el símbolo $, por ejemplo NOMBRE$ es una variable alfanumérica.

A continuación veremos las instrucciones que posee el computador para "dialogar" con el usuario. Estas instrucciones son las que permiten ingresarle datos al computador y que el mismo nos conteste. Las instrucciones son: PRINT e INPUT

Instrucción PRINT

Hace que el computador nos diga algo a través de la pantalla de su televisor. Puede ser una constante o bien entregar el contenido de una variable. La sintaxis es la siguiente:

Para imprimir una constante, la misma debe ir entre comillas:

```
PRINT "HOLA"
PRINT "18"
PRINT "TIENE 18 AÑOS"
```

Son todas construcciones válidas.

El computador no analiza el contenido de lo que se le pide que diga, se podría poner:

```
PRINT "EL SOL SALE DE NOCHE"
```

Y sería perfectamente válido. Para imprimir una variable simplemente hay que poner:

```
PRINT NUMERO
```

Y el computador nos dará el contenido de la variable número. Lo anterior es distinto a imprimir:

```
PRINT "NUMERO"
```

En este caso se va a escribir la palabra número. Las comillas hacen la diferencia, cualquier cosa entre comillas es constante, caso contrario interpreta que es una variable numérica de contenido igual a 0.

Pueden combinarse constantes y variables en una misma instrucción PRINT. Lo mismo se hace utilizando la coma y el punto y coma. En el primero de los casos la impresión se realiza intercalando automáticamente espacios en blanco, entre las variables y las constantes, en el segundo de los casos se imprime todo seguido.

En los ejemplos que vimos anteriormente de las asignaciones y de las operaciones matemáticas,



cuando oprimíamos la tecla RETURN, exteriormente no pasaba nada. Esto es porque no se le pidió al computador que entregue el resultado. Si a continuación de cada una de las asignaciones u operaciones matemáticas hacemos PRINT y el nombre de la variable correspondiente, veremos el resultado de dicha operación. También puede hacerse en forma reducida lo siguiente:

```
PRINT A+B*5
```

Y obtendremos su resultado.

Instrucción INPUT

Hace que podamos ingresarle datos al computador. Dichos datos serán asignados a una variable. Por ejemplo:

```
INPUT EDAD
```

Al ejecutar esta instrucción, aparecerá en pantalla un signo de interrogación, requiriéndonos que ingresemos, por el teclado, el contenido de la variable EDAD. Para responder bastará ingresar el número que deseemos y oprimir la tecla RETURN. Si a continuación se hiciera PRINT EDAD, veríamos que se imprime en pantalla el número previamente ingresado.

Instrucción IF...THEN...

Esta es la instrucción que corresponde al rombo de los diagramas de flujo, es decir la que implica la pregunta y toma de decisión. Su sintaxis es la siguiente:

```
IF condición THEN acción
```

Cuya traducción sería: SI pasa algo ENTONCES hacer tal cosa. Esto se ve claramente con un ejemplo:

```
IF EDAD= 18  THEN PRINT
"tiene 18 años".
```

Lo que estamos haciendo es que si el contenido de la variable EDAD coincide con la constante 18, entonces se debe imprimir el texto "tiene 18 años".

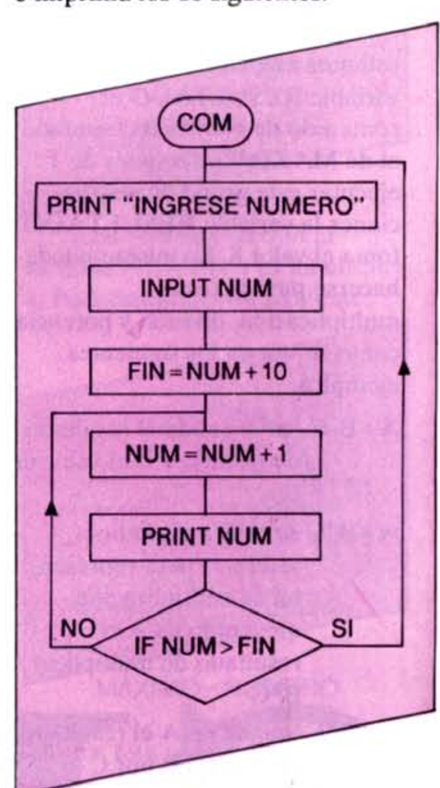
Hasta ahora sólo hemos visto instrucciones en el modo inmediato, o sea que ingresábamos una instrucción y al oprimir la tecla RETURN veíamos su ejecución. Para combinar instrucciones, se utiliza el modo diferido de programa. Para diferenciarlo del anterior se debe poner un número antes de cada instrucción, que cumple ademas, la función de establecer un orden de ejecución entre las instrucciones y la de ponerle un "rótulo" a dicha instrucción. Esto ultimo significa que cualquier instrucción puede ser reconocida por un número. Ese número puede ser cualquiera entre 1 y 32767, pero se acostumbra a trabajar con números de 10 en 10 comenzando del número 100. Esto permite que si en algún lugar deseamos insertar un grupo o simplemente una instrucción, podamos hacerlo.

Instrucción GOTO

La traducción de esta instrucción es IR A. Va acompañada de un número de instrucción y es utilizada para alterar el orden cronológico de las instrucciones. A continuación de esta instrucción, en lugar de ejecutarse la que le sigue, el programa continuará a partir del número de instrucción indicada. Por ejemplo: GOTO 100

Ahora estamos en condiciones de hacer nuestro primer programa. Lo acompañaremos con su correspondiente diagrama. Nos acostumbraremos a hacerlo desde el principio, aunque en el comienzo no le veamos sentido, posteriormente nos será de gran ayuda.

Haremos un primer programa que nos solicite un número e imprima los 10 siguientes.



EL programa correspondiente sería:

```
100 PRINT "INGRESE NUMERO"
110 INPUT NUM
120 FIN=NUM+10
130 NUM=NUM+1
140 PRINT NUM
150 IF NUM>FIN THEN GOTO 100
160 GOTO 130
```

Este programa no termina nunca. Esto se puede comprobar fácilmente, ya que el diagrama correspondiente, no tiene ningún símbolo fin. Para pararlo hay que apretar la tecla BREAK.

En la línea 100 ponemos un texto aclaratorio para que cuando el computador ponga el signo de pregunta sepamos que nos está solicitando. La instrucción 110 hace que en la variable NUM almacene el número desde el cual empezara a contar los diez siguientes. En la línea 120 guardamos en FIN hasta donde queremos que cuente, o sea hasta NUM mas diez. En la siguiente le sumamos uno a NUM para imprimir en la 140 el número correspondiente. En la 150 preguntamos si NUM ya pasó a FIN. Si lo hizo vamos a la línea 100 para comenzar de nuevo desde el principio. Caso contrario volvemos a la línea 130 para sumarle uno y así seguir.

Para volver a ver el programa de nuevo desde el principio, utilizaremos la instrucción LIST en modo inmediato con lo que podremos volver a ver el programa desde el principio.

A continuación estamos en condiciones de programar nuestro primer juego. El mismo se llama adivina el número. Antes aprenderemos la función matemática de generación de números al azar y la de truncamiento de éstos. Las mismas son RND(0) e INT. La primera genera números al azar entre 0 y 1 y la segunda le quita los decimales a los números fraccionarios. También verás que se puede escribir mas de una instrucción por número de línea. Para hacerlo deberás separarlas con el singo dos puntos (:) Veamos el diagrama de flujo.

El programa sería:

```
100 AZAR = INT(RND(0)*100) + 1
110 PRINT"INGRESE NUMERO"
120 INPUT NUMERO
130 IF NUMERO > AZAR THEN
    PRINT "NUMERO MUY
    GRANDE":GOTO 110
140 IF NUMERO < AZAR THEN
    PRINT "NUMERO MUY
    CHICO":GOTO 110
150 PRINT "TE FELICITO HAS
    GANADO"
160 GOTO 100
```



COM

AZAR = INT(RND(0)*100) + 1

PRINT "INGRESE NUMERO"

INPUT NUMERO

NUMERO > AZAR — SI → PRINT "NUMERO MUY GRANDE"

NO

NUMERO < AZAR — SI → PRINT "NUMERO MUY CHICO"

NO

PRINT "TE FELICITO, HAS GANADO"



PROG

PARA

BASIC

En la instrucción 100 le asignamos a AZAR un número de 1 a 100. Como la función RND(0) genera números entre 0 y 1, debemos valernos de esa fórmula para que el rango de valores sea de 1 a 100. El RND(0) multiplicado por 100 da valores decimales entre 0 y 99. Primero los convertimos en enteros con la Función INT y luego le sumamos 1. En las instrucciones 110 y 120, ingresamos nuestras opciones. En las líneas 120 y 130 verificamos los casos de habernos equivocado en mas y en menos. Si no se dan ninguna de las circunstancias anteriores, equivale a decir que hemos acertado, por lo cual imprimimos en la instrucción 150 el mensaje correspondiente y volvemos a comenzar. Este programa tampoco tiene fin. Para detenerlo recuerda oprimir la tecla BREAK.

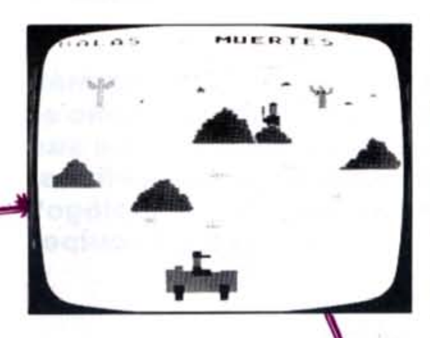
Te desafiamos a que mejores el programa, incorporándole un contador que nos indique en cuantas alternativas dimos con la respuesta correcta.

Ya te hemos dado armas suficientes como para poder hacer tu primer juego. Practica con estas instrucciones y en la siguiente entrega, continuaremos en el fascinante mundo de la programación.
Hasta pronto!!!.

Como habíamos prometido en el número anterior, seguimos entregándoles las novedades que la empresa M.P.M. S.A., a través de su línea de productos TURBO SOFTWARE, continúa lanzando al mercado para la pistola Atari. En este caso estamos presentando otros dos juegos, Emboscada en el desierto y Simon, y un educativo, Laser Math.

RAMAS

LA PISTOLA







Emboscada en el Desierto

En este juego, Ud. se encuentra cubriendo a un contingente armado que se desplaza por el desierto en terreno hostil. El convoy de jeeps, debe atravesar una zona pedregosa cubierta de franco tiradores. Su misión será detectarlos y eliminarlos antes de que puedan llegar a disparar contra sus compañeros.

Una vez que el convoy ha podido pasar con éxito por el desierto, le serán asignadas otras misiones, que también consistirán en cubrir a sus compañeros, pero de enemigos mejor entrenados y armados, por lo que su velocidad de acción y decisión sera exigida al máximo.

Su manejo es muy sencillo y se realiza enteramente con la pistola. Disparando podra hacer que comience el juego o que el mismo continúe, una vez cumplida una misión.

Simon

Este es el tradicional juego del SIMON. En el mismo, se debe repetir la melodía que el computador va eligiendo al azar. La misma se va haciendo cada vez mas larga, incorporando de a una nota a medida que pasan las rondas.

El sonido va acompañado de cuatro figuras de colores, cada una de ellas relacionadas con uno de los tonos que van apareciendo. Para seguir la melodía deberá disparar contra cada una de dichas figuras, según corresponda, para que la melodía sea la correcta. Se puede seleccionar la cantidad de rondas, es decir la cantidad de notas que se debe responder correctamente para ganar.

El tiempo que se demore en responder la melodía, es esencial para obtener un puntaje mayor, existiendo un tiempo máximo para responder.

Este juego puede también ser utilizado con el lápiz de luz: TURBO LIGHT-PEN o bien simplemente con el teclado.

Laser Math

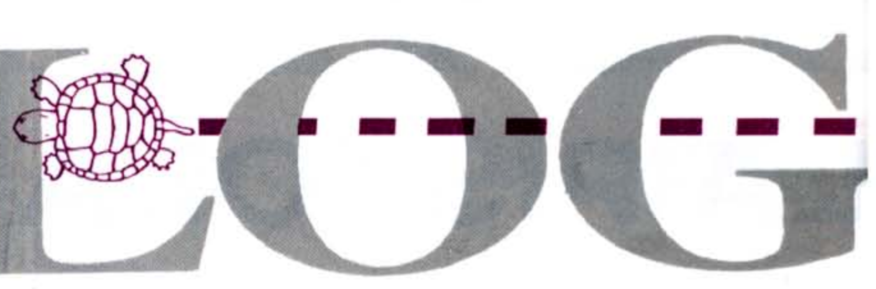
Este educativo, presentado en forma de juego, esta orientado a los niños que se están iniciando en las operaciones matemáticas básicas: suma, resta, multiplicación y división. En el mismo Ud. deberá defender una nave espacial que esta siendo atacada por misiles enemigos.

Para poder destruir los misiles deberá responder correctamente la operación matemática que aparece en el centro de la nave. Le aparecerán cuatro posibles resultados en cada una de las esquinas de su televisor. Disparando con su pistola contra la opción correcta logrará que el proyectil enemigo sea destruido. Escogiendo una opción equivocada hará que el misil se acerque más rápidamente.

Existen cuatro niveles, relacionados con la velocidad con que se acerca dicho misil.

Este educativo puede también ser utilizado con el lápiz de luz: TURBO LIGHT-PEN o bien simplemente con el teclado.

APRENDIENDO A



El Logo es un lenguaje diseñador para los más pequeños. Con él, el niño puede ver cómo el computador va ir respondiendo a sus instrucciones, en la realización de gráficos, estableciéndose así un primer "diálogo" con su equipo.

El logo cuenta con cuatro lápices para trabajar, cada uno de los cuales posee un color y una forma determinada. La famosa "Tortuga" del logo es la forma que viene predeterminada. Tanto el color como dicha forma son modificables, según los deseos del dibujante.

Comenzaremos a ver algunas instrucciones del Logo con las cuales podrás realizar tus primeros gráficos.

Debemos antes mencionar, que la pantalla del Logo Atari puede dividirse en dos. La parte dedicada a gráficos y la parte dedicada a instrucciones. Ambas pueden ser vistas en forma simultánea o por separado. Lo primero es recomendable para ir viendo cuál es el resultado de cada orden.

El control de estas opciones se logra oprimiendo simultáneamente las siguientes teclas:

Para ver sólo el gráfico: oprimir teclas CONTROL y letra F.

Para ver sólo las instrucciones: oprimir CONTROL y la letra T.

Para ver gráficos e instrucciones: oprimir CONTROL y la letra S.

Ahora si podemos comenzar a ver las instrucciones. Para probarlas, conviene primero convertir la pantalla, que viene sólo para texto, en la combinación Texto Gráfico, oprimiendo las teclas CONTROL y S.

Después de ingresar las instrucciones, para que comiencen a funcionar, hay que oprimir la tecla RETURN.

Nota: Para poder usar su computador con el lenguaje logo, deberá adquirirlo en el mercado.

Instrucción DILE

Como vimos anteriormente el logo posee cuatro tortugas numeradas del 0 al 3. Con la instrucción DILE, se le indica al computador cuál o cuáles de las cuatro tortugas deberán obedecer las instrucciones que le sigan.

Si la instrucción está dirigida a más de una tortuga la lista de las mismas debe ir entre corchetes.

Ejemplos:

DILE 0

Todo lo que hagamos a continuación, será obedecido por la **"tortuga 0"**.

DILE [0 1 3]

Todo lo que hagamos a continuación, será obedecido por las tortugas 0, 1 y 3. Puede elegir cualquier combinación de las cuatro tortugas.

Instrucción AV

Su significado es "avanza". Con esta instrucción la o las tortugas que se encuentren activas por las instrucción DILE avanzará tantos pasos como lo indique el número que la acompaña.

Ejemplo:

AV 20

La o las tortugas indicadas por la última instrucción DILE avanzarán 20 pasos.

Instrucción RE

Su traducción es "retrocede". Es la instrucción contraria a AV. La o las tortugas activas, retroceden tantos pasos como los indicados por el número que le sigue.

Ejemplo:

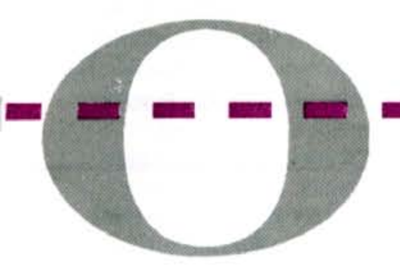
RE 20

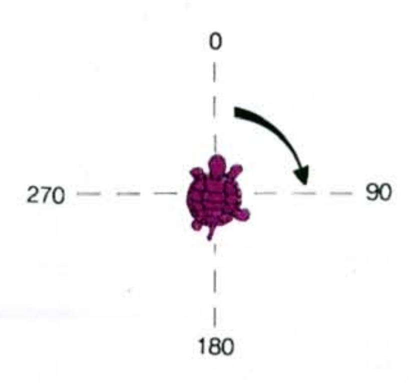
La o las tortugas activas retroceden 20 pasos.

Instrucción DE

Su significado es "derecha". Las instrucciones que se encuentren activas girarán a la derecha la cantidad de grados que indique el número que la acompaña. Este número puede ser de 0 a 360. En el siguiente gráfico verás cuánto gira la tortuga según este número.

DIBUJAR CON





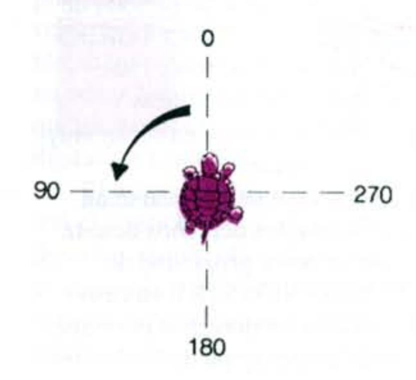
Ejemplo:

DE 180

Las tortugas activas girarán a la derecha 180 grados, por lo que quedarán mirando exactamente al lado contrario que miraban.

Instrucción IZ

Su significado es "izquierda". Es exactamente igual que DE pero a la izquierda. El gráfico te muestra el movimiento de la tortuga dependiendo del valor.



Ejemplo:

IZ 180

Nótese que, en este caso, el efecto es el mismo que el de DE 180, pero lo hizo hacia la izquierda.

Instrucción LA

Su significado es "lápiz activado". Se utiliza para que cuando la tortuga avance o retroceda en la dirección indicada por DE e IZ, vaya dejando una línea, que va a formar parte del dibujo.

Ejemplo

LA

Instrucción SL

Su significado es "sin lápiz". Es lo contrario a LA (lápiz activado), es decir que todos los movimientos que se realicen, serán hechos sin dejar trazos a su espalda.

Instrucción LG

Su significado es "lápiz goma". A partir de la ejecución de esta instrucción, la o las tortugas activas irán borrando todas las líneas que encuentren a su paso cuando se ejecuten las instrucciones de avanzar o retroceder. Se utiliza como goma de borrar.

Instrucción LM

Su significado es "limpia la pantalla". Luego de ejecutada esta instrucción la pantalla de gráficos se borrará retornando la tortuga a su posición original.

LM

Verás como todo lo que has hecho se borra, volviendo la tortuga a su posición original, es decir el centro.

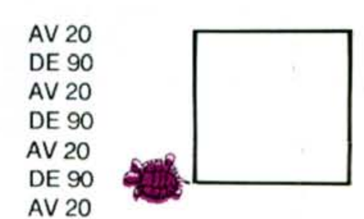
Instrucción BORRA

Es similar a LM. La única diferencia es que la tortuga no vuelve a su posición inicial.

En el siguiente número continuaremos con mas instrucciones. A modo de ejemplo, a continuación verás las instrucciones para hacer un cuadrado y una silla. Tómalos como base para hacer figuras más complicadas.

CUADRADO

AV 20
DE 90
AV 20
DE 90
AV 20
DE 90
AV 20



SILLA

AV 30
RE 15
DE 90
AV 15
DE 90
AV 15



Interrupciones de Display List

En esta oportunidad continuaremos con el despliegue de pantalla, por la importancia que tiene en la presentación de un programa.

Como vimos en el número pasado de nuestra revista, ANTIC es un microprocesador y como tal es programable, ésto nos permite la generación de una pantalla a nuestro entero gusto, usando cualquiera de los 16 modos gráficos disponibles. Incluso podríamos crear una lista de despliegue que contenga todos los modos gráficos en una sola pantalla.

Pero eso no es todo, ya que ANTIC tiene la posibilidad de comunicarse directamente con el 6502 a través de la línea de interrupción no enmascarable (NMI), lo que nos permite detener al procesador para efectuar una operación en cualquier momento durante el despliegue de una pantalla, y luego puede continuar con la función que se estaba ejecutando. Las posibilidades de esta función, son en términos de trabajo con gráficos, casi ilimitadas. Por ejemplo podemos cambiar colores, mover y subdividir players, cambiar caracteres, o ejecutar cualquier función que pueda realizar el 6502.

Este método de interrupción comienza con ANTIC, que al encontrar en la lista de despliegue el bit de interrupción puesto a uno (bit 7) se informa que debe interrumpir al 6502, para esto espera a que termine el despliege de la línea de modo gráfico que está trazando, y luego verifica que en el registro de NMIEN (interrupción no enmascarable habilitada) se encuentre autorizada su

JUEGOS

A partir de este número incluiremos en nuestras páginas como mínimo un juego en cada edición de tu revista TURBO NEWS (R), permitiéndote así aprender con nuestros cursos y disfrutar de excelentes juegos, como el que a continuación describiremos. (Todo en una sola revista!).

Por ser éste el primer juego que publicamos lo hemos elegido por sus cualidades de entretención-aprendizaje, pero si tienes alguna preferencia especial por algún tipo de juego, esperamos tu carta.

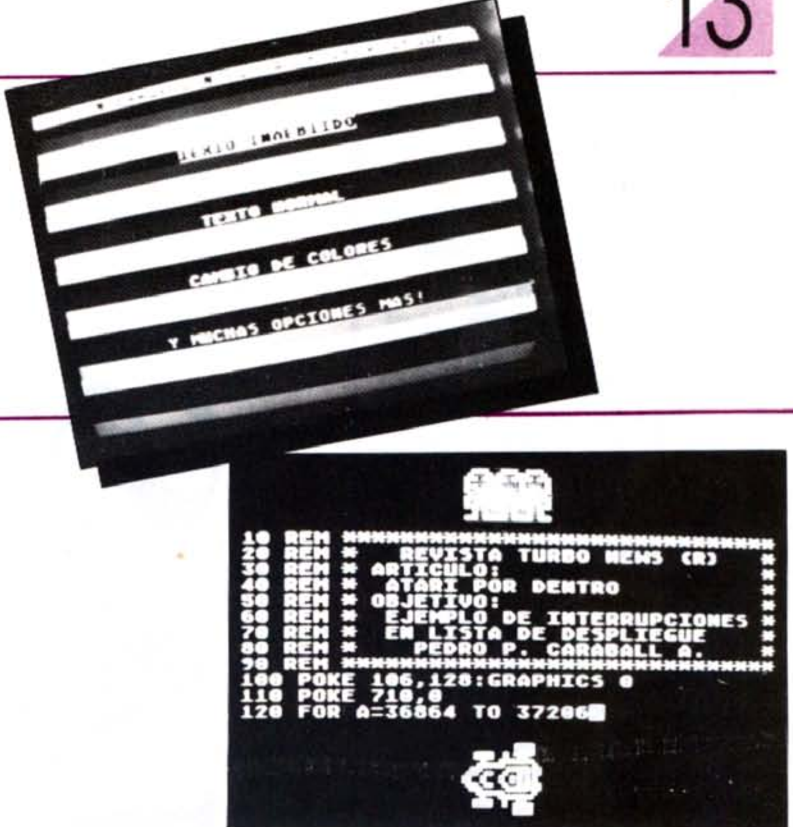
Si deseas enviar algún juego o programa en general para su publicación, podras ganar una subscripción por un período de seis meses a tu revista TURBO NEWS (R). Esperamos tu colaboración (las críticas y correcciones también serán muy bien recibidas).

En caso de publicarse un programa, los derechos de este pasarán a ser propiedad de TURBO NEWS (R), en caso contrario los derechos permanecerán como propiedad del autor.

interrupción (bit 7 de $D40E), si el bit correspondiente está puesto a uno continúa bajando la línea de NMI del 6502. A partir de este momento ANTIC continúa con sus funciones habituales. El 6502 toma control del proceso en este momento a través del vector de interrupción de lista de despliegue (VDSLST que se encuentra en $200 y $201),para retornar a la función que el 6502 estaba realizando antes de ser interrumpido debemos usar la instrucción RTI (retorno de interrupción), debemos tener cuidado con los registros ya que el sistema operativo no se ocupa de guardarlos para esta interrupción.

Este ciclo se repite para cada pantalla desplegada por ANTIC.

En la sección de programas entregamos un ejemplo de interrupción de lista de despliegue, en la cual los primeros 3 players son divididos, toman una distinta dirección y un distinto color.



Este primer programa que estamos publicando,es un juego destinado a quienes esperamos entregar el mejor material, los mas chicos. Para que jueguen reconociendo letras y números. Para los mas grandes, que ya hacen algunos programas en BASIC creemos que las rutinas aquí contenidas pueden ser de mucha utilidad para sus propios programas.

Este programa esta escrito en BASIC con rutinas en lenguaje de máquina, para que tenga una presentación atractiva y una velocidad de ejecución considerable, asegurando así un buen juego en pocas líneas de programa, lo que se traduce en facilidad para digitación y mucha entretención.

El juego consiste en buscar con el joystick, moviéndolo a la derecha o izquierda, la letra que es igual a la grande desplegada en la parte superior de la pantalla, mientras más rápido encuentres la letra igual serás recompensado con más puntos, si por el contrario te equivocaras al elegir, se te descontarán bonos que no serán sumados al puntaje cuando encuentres la letra correcta. Esto te será avisado con un sonido.

Esperamos que este programa te de entretención y aprendizaje. Y te desafiamos a incorporar el display numérico para mostrar los tiempos de respuesta.



Una breve descripción de la operativa del programa:

LINEA 100: modo gráfico, colores y margen izquierdo.

LINEA 120: dirección lista de despliegue

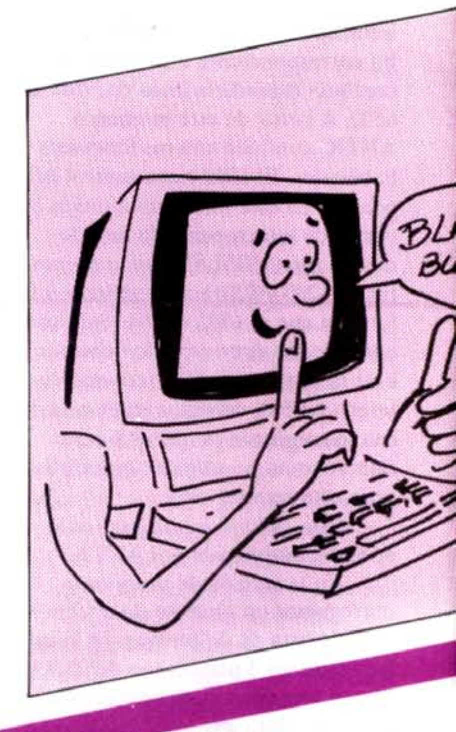
LINEA 130: permite una interrupción en la lista de despliegue

LINEA 140: rutina en lenguaje de máquina para interrupción (ver listado informativo)

JUEGOS

LINEA 150: rutina en lenguaje de máquina para copiar un carácter desde el ROM a gráfico (ver listado informativo)
LINEAS 160 Y 170: apuntan vector de interrupción a la rutina creada para este efecto
LINEA 180: crea un carácter vacío y uno lleno para despliegue en gráfico 2
LINEA 182: permite la interrupción de lista de despliegue y cambia el set de caracteres
LINEA 184: presentación (conteo regresivo)
LINEA 190: fin presentación
LINEAS 191,192,195: impresión textos
LINEA 200: elige carácter al azar y compara que no sea igual al anterior
LINEA 210: muestra carácter grande y determina tiempo de bonos
LINEA 215: espera que el botón del joystick no este presionado
LINEA 220: movimiento del joystick
LINEA 230: mueve el cursor con sonido
LINEA 240: espera a que el botón del joystick esté presionado
LINEA 260: toma carácter bajo el cursor
LINEA 270: carácter chico igual al grande?
LINEA 280: error, el carácter elegido no es igual al grande
LINEA 300: cálculo bonos, despliegue de puntaje y sonido
LINEA 310: comience con otro carácter
LINEAS 500,510,520: subrutina de espera y silencio en sonido





En esta columna entregaremos habitualmente, rutinas y programas utilitarios de uso general, que esperamos sirvan para complementar los programas escritos por ustedes mismos.

Para nuestra primera edición de rutinas y programas utilitarios, hemos elegido sólo rutinas de uso general, para asegurar que al menos una de ellas sea útil en alguno de tus programas, ya sea aumentando su velocidad o facilitando su construcción.

Si tienes alguna duda o pregunta de algo que tu solo no puedes resolver, o no tienes la información suficiente como para hacerlo, la solución está en tus manos, sólo escríbenos y tu respuesta no se hará esperar, aparecerá publicada en estas páginas.



La primera rutina está escrita en lenguaje de máquina, y cumple la función de recomenzar la ejecución del programa basic que se encuentre en memoria, si es presionada la tecla RESET. Esta rutina debe ser la primera línea del programa basic, así si quieres que tu programa no se detenga, debes copiar la línea número 0 del listado correspondiente en tu propio programa, las demás líneas sólo se entregan como una demostración. Aunque la línea 0 se modifica a si misma, puedes grabar y cargar tu programa sin dificultad alguna.

La segunda rutina es sin lugar a dudas una de las mejores herramientas de trabajo que podrás tener para el traspaso de información desde o hacia un periférico. La misma te permite el traspaso de gran cantidad de información en un tiempo muy inferior al que tomaría el BASIC para el mismo trabajo. Para incorporar esta rutina en tus programas deberás copiar la línea número 10 del programa respectivo.

Para hacer un buen uso de esta rutina deberás seguir estas sencillas instrucciones de uso.

Llamada a la rutina:

X=USR(CIOV,CANAL, OPERACION,DIRECCION,CANTIDAD)

Canal: depende del canal abierto para el trapaso de información, así si usted ejecutó "OPEN #5,4,0,A$" el canal correspondería al número 5.

Operación: usando el ejemplo anterior, el canal ha sido abierto para lectura, correspondiendo la operación en este caso al número 4. Este valor coincide con los de entrada y salida normales.

Dirección: es la posición de memoria que contiene la información, o la que va a recibirla, según se trate de una lectura o escritura.

Cantidad: es la cantidad de bytes a ser traspasados.

Desde el basic podemos hacer este sencillo ejemplo comparativo de tiempos de ejecución para formarnos una idea de lo poderosa que es esta rutina. (este ejemplo copia el basic a una pantalla de gráfico 8).

Programa normal en basic:

```
10 OPEN #1,8,8,"S:"
20 FOR A=40960 TO 49151
30 PUT #1,PEEK(A)
40 NEXT A
50 CLOSE #1
60 POKE 712,PEEK(20):GOTO 60
```

Programa con la rutina ciov incorporada:

```
Copiar la rutina ciov en la línea 10
20 OPEN #1,8,8,0,"S:"
30 X = USR(CIOV,1,8,40960,8192)
40 CLOSE #1
50 POKE 712,PEEK(20):GOTO 50
```

Al comenzar el cambio de color en el borde, la transferencia está completa.

La rutina ciov retorna al basic con la cantidad de bytes que fueron efectivamente transferidos sin ningún error.

La tercera y última rutina publicada en este número está reservada a los usuarios que poseen unidad de disco (cualquier modelo). La rutina en si, es similar a la rutina ciov, sólo que en este caso no es necesario abrir un canal para la intercomunicación, con la consiguiente ventaja que ello implica.

El uso apropiado de esta rutina es como sigue:

X=USR(SECTOR,NUMSECTOR, COMANDO,DIRECCION,DRIVE)

Numsector: es el número del sector que queremos leer o escribir.

Comando: es la operación que vamos a efectuar.

ASC("R")= lectura
ASC("P")= escritura
ASC("W")= escritura con verificación

Dirección: es la posición de memoria desde o hacia donde se va a efectuar la transferencia.

Drive: es el número de identificación de su unidad de disco (1,2,3 o 4)

En X es retornado el estado de la operación, así si el resultado de la operación es exitoso X sera igual a 1, en caso de algún problema, X contendrá el número de error que se ha producido.

Esperamos que este primer aporte a los programadores sea muy bien aprovechado por ustedes.

Display Nu

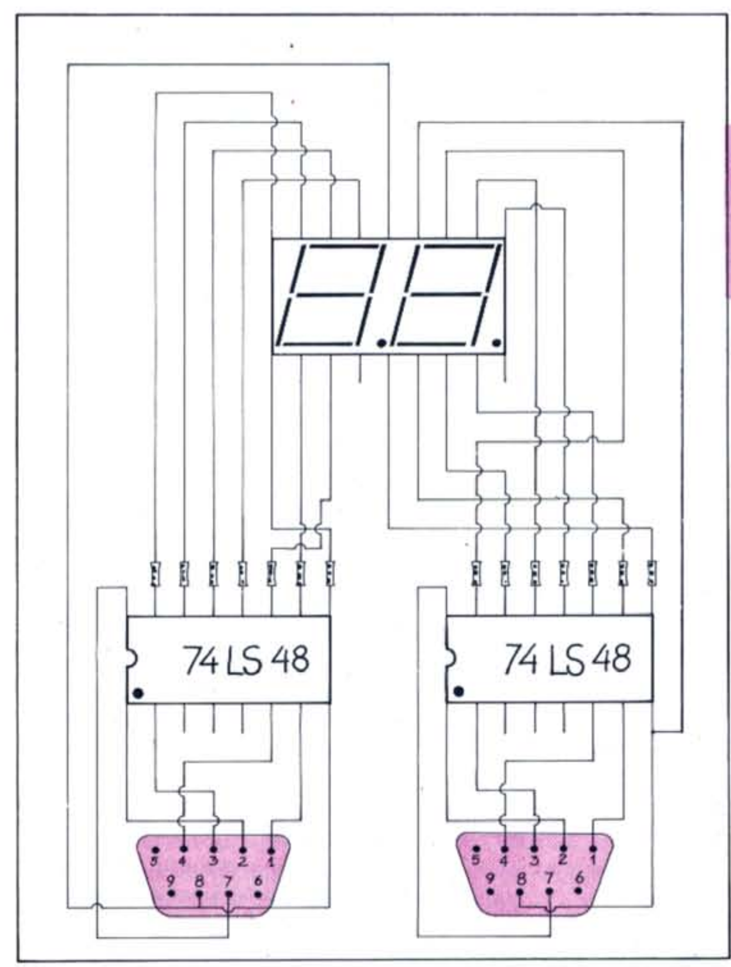
El hardware de este mes corresponde a un display numérico el cual puede emplearse en múltiples funciones. Para la construcción de este circuito es recomendable analizar el funcionamiento de los enchufes de los joysticks, en cualquier manual de su computador o en este mismo artículo del número anterior de su revista TURBO NEWS (R).

Lista de materiales (dos dígitos):

2 enchufes de joysticks
2 cables de 6 conductores
2 circuitos integrados 74LS48
14 resistencias de 270 ohms
1 display tipo DC56-11HDB
1 placa apropiada para el montaje y conexión.

Los desarrollos de hardware que estamos presentando son fabricados con materiales sencillos, baratos y de fácil construcción. Además no representan ningún riesgo ya sea para el computador o para quienes trabajen en ellos. Tratando así que grandes y chicos se interesen y puedan construir circuitos electrónicos que van a proporcionar a fin de cuentas entretención y aprendizaje.





Circuito dos dígitos

Circuito un d

Lista de materiales (un dígito):

1 enchufe para joystick
1 cable de 6 conductores
1 circuito integrado 7447
7 resistencias de 270 ohms
1 display tipo FND507
1 placa apropiada para el mo
y conexión

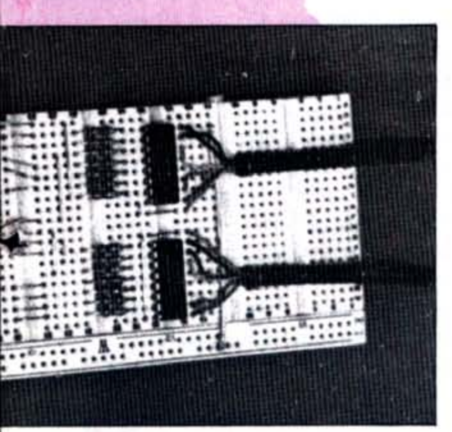
nérico

Para el correcto funcionamiento de este display se debe tener en consideración la siguiente tabla.

N= número desplegado
B= valor para joystick B
A= valor para joystick A
7= bit 7 valor 128
6= bit 6 valor 64
5= bit 5 valor 32
4= bit 4 valor 16
3= bit 3 valor 8
2= bit 2 valor 4
1= bit 1 valor 2
0= bit 0 valor 1

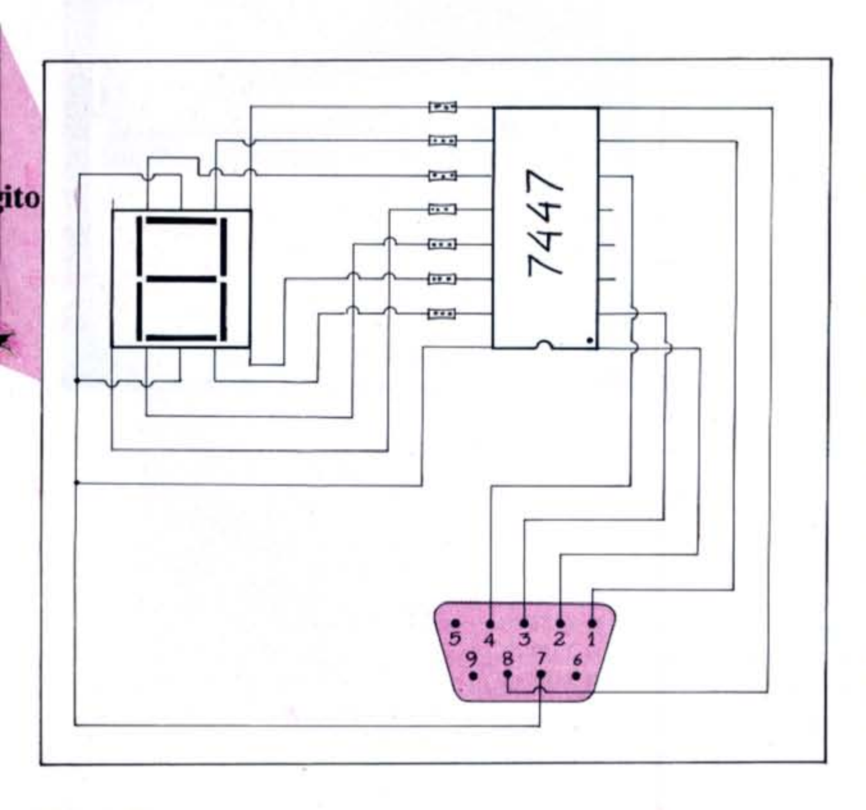
PORT A ($D300)

Enchufe B						Enchufe A				
N	B	7	6	5	4	3	2	1	0	A
0	0	0	0	0	0	0	0	0	0	0
1	16	0	0	0	1	0	0	0	1	1
2	32	0	0	1	0	0	0	1	0	2
3	48	0	0	1	1	0	0	1	1	3
4	64	0	1	0	0	0	1	0	0	4
5	65	0	1	0	1	0	1	0	1	5
6	96	0	1	1	0	0	1	1	0	6
7	112	0	1	1	1	0	1	1	1	7
8	128	1	0	0	0	1	0	0	0	8
9	129	1	0	0	1	1	0	0	1	9
OFF	240	1	1	1	1	1	1	1	1	15



En esta foto observamos la construcción del circuito de display para dos dígitos, montado en una placa para prototipos (protoboard)

gito



Por ejemplo si usted quisiera que el número de las decenas mostrara un tres y el de las unidades un seis, deberá sumar el valor que tenga B para el número tres, en este caso corresponde un 48 con el valor que tenga A para el seis, en este caso un 6. Este resultado debe reflejarse en la posición de memoria $D300 (54016), ésto se hace mediante POKE 54016,48+6

Para programar los enchufes de joysticks como salidas se deben ejecutar las siguientes instrucciones: POKE 54018,48:POKE 54016,255:POKE 54018,60.

En caso de tener un solo dígito para usar, usted puede elegir cuál de los dos enchufes va a usar como salida siguiendo el ejemplo que se entrega a continuación.

Para usar el enchufe B (posterior): POKE 54018,48:POKE 54016,240:POKE 54018,60

Para usar el enchufe A (anterior): POKE 54018,48:POKE 54016,15:POKE 54018,60

Se debe tener en cuenta cuál de los dos enchufes se va a utilizar para depositar los valores correctos en $D300 (54016), según la tabla anterior.

En nuestro listado de programas usted puede encontrar un ejemplo de uso para este display numérico. En este programa hay dos líneas que son las más importantes y las únicas que tendremos en cuenta para nuestra explicación.

Línea 100: programa los dos enchufes de joysticks como salidas.

Línea 1000: es un cálculo numérico en el cual se convierte el número pasado en la variable N, en la representación interna necesaria para desplegar el número N en el display numérico.

RANKING DEL MES

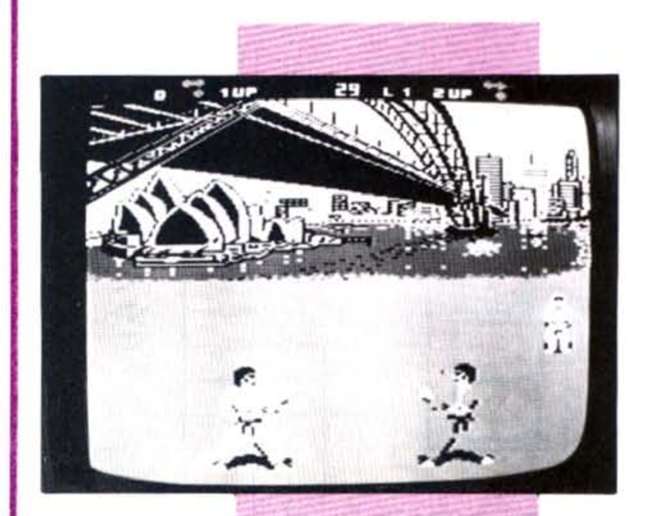
Pos. del mes		Pos. mes anterior	Título
1		2	NINJA
2		5	INTERNATIONAL KARATE
3		4	MONTEZUMA'S REVENGE
4		1	SCREEMING WINGS 1942
5		-	FUTBOL II
6		-	MIRAX FORCE
7		3	BRUCE LEE
8		8	POLE POSITION
9		14	GREAT AMERICAN RACE
10		10	RIVER RAID
11		-	HARDBALL
12		6	LEADER BOARD GOLF
13		15	BLUE MAX
14		7	MONTEZUMA
15		19	FLAK
16		18	ELEKTRAGLIDE
17		-	FIGHTER PILOT II
18		24	NINJA MASTER
19		21	BOULDER DASH II
20		9	ALLEY CAT
21		12	BMX SIMULATOR
22		20	BOINA VERDE
23		-	ZAXXON
24		25	STAR RAIDERS II
25		-	CRYSTAL RAIDER

Este es el ranking correspondiente al mes de Junio, obtenido en base a las estadísticas de ventas de cassettes Turbo Software en todo Chile. Recuerda que tus preferencias también serán tenidas en cuenta, para lo cual podrás escribir a Holanda 2456, Providencia, con los juegos de tu elección.

A continuación encontrarás una descripción de los juegos que van a ir ocupando los primeros lugares. La descripción del NINJA y del SCREEMING WINGS 1942 ya han sido entregadas en el número anterior, por lo que se explicarán los primeros tres excluidos aquellos.

International Karate

En este excelente juego de artes marciales, Ud. controla un karateka, y deberá enfrentar a su oponente en una lucha a muerte, por los distintos escenarios que le irán apareciendo. Ud. cuenta con dieciséis movimientos distintos compuestos por saltos, patadas, golpes y demás artificios que componen esta disciplina. Los mismos podrán ser realizados moviendo el joystick en sus ocho posiciones posibles (izquierda, derecha, arriba, abajo, y cuatro diagonales) más los mismos ocho con el botón del joystick apretado. La efectividad de sus golpes, no sólo su frecuencia, serán fundamentales para el desarrollo de la lucha. Podrá jugar contra el computador o contra otro adversario, mediante la utilización de un segundo joystick. Cada vez que logre eliminar a su adversario, será sometido a una dura prueba que, en caso de ser superada, hará que le sea otorgado un nuevo cinturón.



SONIDO	6
GRAFICACION	7
ADICCION	6.9
PRESENTACION	7
PROMEDIO	6.725

Montezuma Revenge

SONIDO	6.5
GRAFICACION	6.8
ADICCION	7
PRESENTACION	6.8
PROMEDIO	6.775



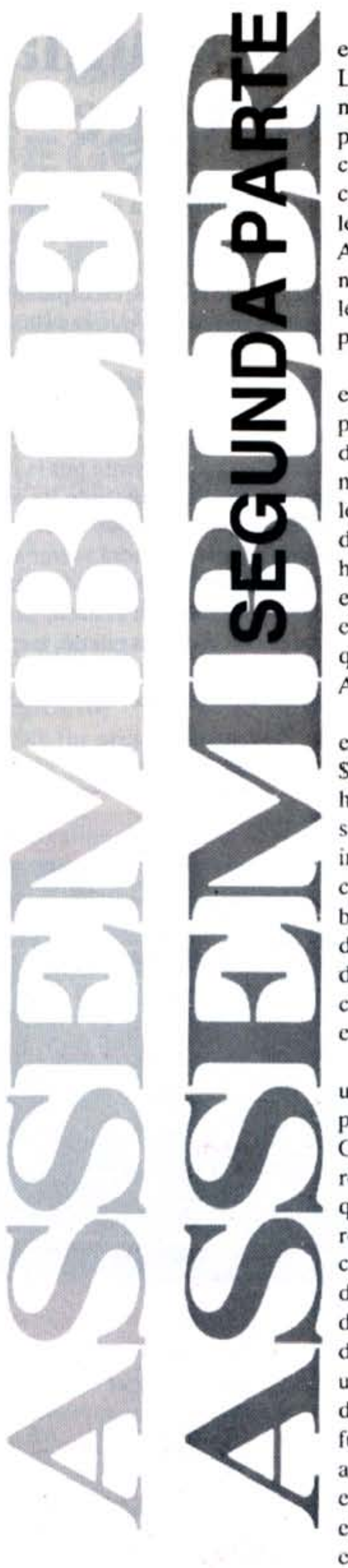
Ayude al valiente explorador a ingresar en las profundidades de la tumba del emperador Montezuma, en búsqueda de los tesoros perdidos. Los lúgubres pasillos están infestados de víboras, escorpiones y muchas otras alimañas que fueron puestas allí para defender las riquezas imperiales. En el recorrido, además de las joyas, deberá ir recogiendo las llaves que le permitirán ir franqueando las pesadas puertas de hierro. Cada una de dichas puertas posee un color determinado y será abierta únicamente por la llave del mismo color. También deberá ir tomando las espadas que le permitirán defenderse, teniendo en cuenta que una vez utilizada una espada no podrá volver a usarse, y las antorchas, para ir alumbrando el accidentado camino. Su estado físico deberá ser el ideal, ya que será la única manera de poder saltar las grietas, trepar las sogas y esquivar los distintos peligros que deberá enfrentar. Un excelente juego para todas las edades.

Fútbol II

Un clásico partido de fútbol para jugar de a una, enfrentando al computador, o dos personas, controlando cada uno, sus once jugadores, con toda la emoción de los grandes eventos deportivos. Realice las mejores jugadas, pases milimétricos, espectaculares gambetas y veloces arremetidas al arco rival, para poder llegar así al gol. Para poder dirigir la pelota deberá apuntar su joystick a la dirección deseada y oprimir al botón del mismo. El jugador que Ud. controla, dentro de su equipo, es aquel de color más claro, el resto, tiene movimiento propio. Dicho jugador se remarca automáticamente y es aquel que está más cercano a la pelota. Al realizar un pase, el jugador que lo recibe pasa a ser el de color claro, mientras que el que venía conduciendo la jugada retoma el color correspondiente a su equipo.



SONIDO	6
GRAFICACION	6.3
ADICCION	6.6
PRESENTACION	6.5
PROMEDIO	6.35



Uno de los problemas para empezar a aprender a programar en Lenguaje Assembler es que la mayoría de los libros disponibles para programar el 6502 son libros de consulta. Estos libros son útiles cuando ya se ha aprendido la base del lenguaje. El propósito del curso de Assembler de la Revista TURBO news es enseñarte a programar en lenguaje assembler para que luego puedas aprovechar dichos libros.

Antes de empezar a programar en Assembler, necesitarás un programa Ensamblador. El ensamblador es un programa que convierte nombres de instrucciones de tres letras, llamados nemónicos, y direcciones de memoria o números hexadecimales, llamados operandos, en números binarios de ocho bits que el microcomputador 6502, que es el que utiliza nuestro Computador Atari, puede interpretar y ejecutar.

Por ejemplo, cada vez que el ensamblador ve una instrucción JMP $A000, lo convierte en los números hexadecimales $4C $00 $A0. El $4C surge de la traducción de la instrucción JMP que en el 6502 se corresponde con el byte $4C. Los bytes $00 $A0 corresponden a la dirección $A000, los cuales están definidos como LO y HI bytes, tal como se almacenan las direcciones en el 6502.

El programa Ensamblador es una herramienta para obtener el programa en lenguaje de máquina. Con el ensamblador, no es necesario recordar todos los números diferentes que conforman, sólo tendrás que recordar los diferentes nombres que componen el juego de instrucciones del 6502. En las ediciones siguientes de TURBO news encontrarás tablas de referencia en las cuales podrás ubicar listadas todas las instrucciones del 6502 con sus características de funcionamiento, para así tener un ayuda memoria. Como lenguaje ensamblador te podemos recomendar el MAC/65, programa que lo encontraras tanto en diskette o cartridge, o bien el Editor de Assembler que también lo podrás obtener en Cassette.

Una vez definida la utilidad del programa ensamblador, vamos a dejar de lado la forma de utilizarlo para ingresar ahora en el lenguaje Assembler, y veremos el funcionamiento del ensamblador o editor a medida que vayamos desarrollando ejemplos de aplicación.

El uso de los registros (A,X,Y):

Imagina un programa en lenguaje Basic, en donde las únicas operaciones disponibles son la suma y la resta. Además, junto a las variables normales usadas en el programa hay tres variables especiales: A, X e Y. El motivo por lo cual estas variables son especiales, es que tienen que estar incluidas siempre en todas las operaciones aritméticas y de asignación. Dicho de otra forma, no se podría usar la sentencia en esta forma:

```
100 MIN = VAL
```

en vez de ésto, es una limitación que tiene el lenguaje assembler, la necesidad de utilizar los registros A, X o Y, en esta forma:

```
100 A = VAL
110 MIN = A
```

o bien utilizando el registro X:

```
100 X = VAL
110 MIN = X
```

y utilizando el registro Y las instrucciones tendrían la siguiente estructura:

```
100 Y = VAL
110 MIN = Y
```

Esta misma situación se presenta en el caso de las operaciones de sumas y restas en las cuales no se puede mantener la estructura:

```
100 CANT = CANT + 1
```

sino que en lenguaje Assembler es necesario seguir los siguientes pasos para incrementar en uno a la variable CANT:

```
100 A = CANT
110 A = A + 1
120 CANT = A
```

Esta es la idea del uso de los registros A, también denominado "ACUMULADOR", y los registros X e Y.

Físicamente un registro es una posición en la cual se pueden almacenar 8 bits. Es parecido a una dirección de memoria, con la excepción de que los registros se encuentran dentro del procesador 6502 y no en la memoria del computador, lo cual acelera su frecuente utilización. Esta estructura de funcionamiento del 6502 a partir de sus tres registros, se debe fundamentalmente a que es mucho más fácil diseñar un procesador que realice operaciones en un número contenido dentro de sus propios registros, que diseñar uno que pudiera trabajar con números ubicados en la memoria.

Una vez comprendido el uso de los registros del computador, analizaremos la utilización de las instrucciones de los registros A, X e Y.





Instrucción "LDA":

Carga el acumulador a partir de la memoria. Esta instrucción transfiere el contenido de una posición de memoria al acumulador del 6502.

LDA

Acumulador ← Memoria

Instrucción "STA":

Almacena el contenido del acumulador en una dirección de la memoria.

STA

Acumulador → Memoria

Instrucción "LDX":

Carga al registro X con el contenido de una posición de la memoria especificada. Es similar a la instrucción "LDA" pero en este caso el registro utilizado es el X.

LDX

Registro X ← Memoria

Instrucción "STX":

Almacena el valor del registro X dentro de una posición de memoria. Esta instrucción actúa como el "STA" pero utiliza en cambio el registro X.

STX

Registro X → Memoria

Instrucción "LDY":

Carga el registro Y con el contenido de una posición de la memoria.

Instrucción "STY":

Almacena el valor del registro Y en una dirección especificada de la memoria.

Instrucción "TAX":

Transfiere el contenido del acumulador al registro X.

Instrucción "TXA":

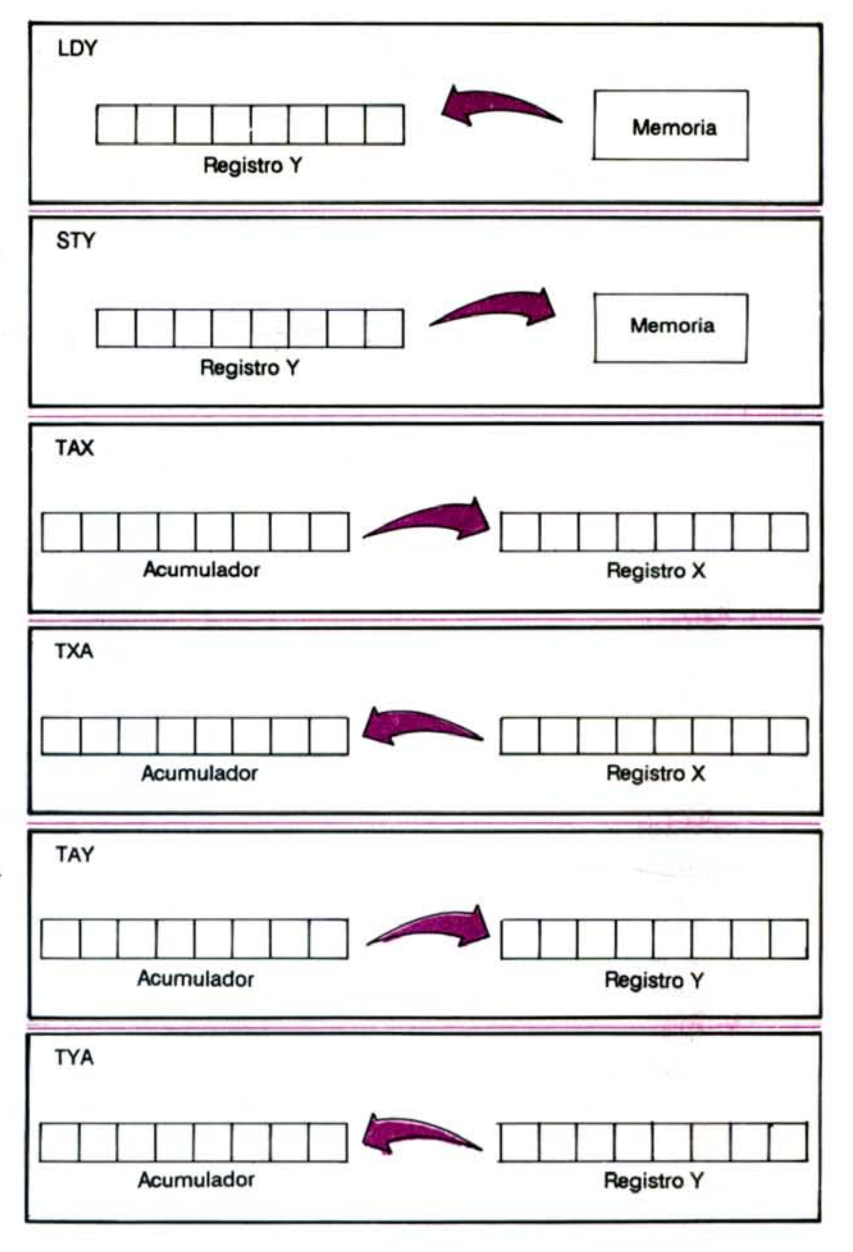
Transfiere el contenido del registro X al Acumulador.

Instrucción "TAY":

Asigna el contenido del acumulador en el registro Y.

Instrucción "TYA":

Define al registro Y con el valor contenido en el Acumulador.



ASSEM

Una vez analizadas las instrucciones de desplazamientos entre los registros, vamos a incorporar en este artículo la primer subrutina en lenguaje Assembler. Para ésto vamos a introducir el concepto de las instrucciones JSR y JMP.

Instrucción "JSR":

La instrucción JSR en Assembler es similar a la instrucción Gosub en el Basic. Cuando el procesador encuentra una instrucción JSR, almacena la posición en donde se encuentra ejecutando el programa, y salta a la dirección ubicada en la instrucción JSR. Si nos encontramos con una instrucción JSR $F2B0 el procesador se dirige a la dirección $F2B0 y ejecuta las instrucciones allí contenidas hasta encontrarse con la instrucción RTS, similar al RETURN del basic. Una vez localizada la instrucción RTS, el procesador vuelve a la instrucción siguiente al "JSR" y continúa con el procesamiento de las próximas líneas.

Instrucción "JMP":

Cuando el procesador se encuentra con una instrucción JMP salta directamente a la dirección por ella indicada, pero a diferencia del JSR, no retorna a la dirección donde fue encontrada. Es similar a la instrucción GOTO en BASIC.

Ahora estamos en condiciones de generar nuestro primer programa en Lenguaje Assembler. Como todavía no hemos analizado todas las instrucciones de este lenguaje, el programa va a ser muy sencillo. Para definirlo, una vez cargado el MAC/65 en el computador, deberás ingresar las siguientes líneas:

```
0100 ;
0110 ;  SUBRUTINA DE IMPRESION DEL
0120 ;  CARACTER A EN LA PANTALLA
0130 ;
0140 ;
0150 ;
0160    .OPT OBJ
0170 ;
0180 ; DIRECCION DE INICIO
0190 ; DE LA SUBRUTINA:
0200 ;
0210    *=  $4000
0220 INICIO
0230    LDA #$41
0240    JSR $F2B0
0250    JMP INICIO
```

Como podrás notar, el programa está constituido por una sucesión de líneas numeradas, tal como un programa en lenguaje Basic. Las instrucciones comprendidas entre las líneas 100 y 150 son comentarios, pues comienzan con el carácter ";". Estas líneas son útiles para mencionar que es lo que se esta haciendo en el programa, para luego recordar cada paso, estando el programa bien documentado.

La línea 160 .OPT OBJ, es una instrucción orientada al Ensamblador. Esta instrucción en el programa en lenguaje de máquina no aparece. Se utiliza para indicarle al Editor que al compilar el programa genere el mismo en lenguaje de máquina, pues en algunos casos podemos compilarlo sin querer que este se genere, como es el caso de analizar determinada instrucción sin necesidad de ejecutar luego el programa. Pero como nosotros deseamos ejecutar este primer programa, le aclaramos al ensamblador que lo genere en lenguaje de máquina.

En la línea 210 con la instrucción * = $4000, le estamos informando que nuestro programa se va a alojar en la dirección $4000 de la memoria. Esta es una ventaja del Lenguaje Assembler, pues el usuario puede definir en qué área de la memoria puede localizar un programa, con algunas restricciones debidas a las direcciones que el computador ya tiene asignadas para su propio uso. En los siguientes números de TURBO news iremos analizando las distintas direcciones que el computador tiene disponible para que definamos nuestros programas.

La línea 220 define solamente una etiqueta o rótulo. Este instrucción que define a INICIO como rótulo es muy útil, pues es mas práctico cuando tenemos que saltar al comienzo del programa, ingresar la línea JMP INICIO que tipear JMP $4000.

En la línea 230, estamos cargando el acumulador con el byte $41. Esta es una forma especial de utilizar la instrucción LDA, pues en vez de cargar al registro A con el contenido de una dirección de la memoria, carga al acumulador con el byte que sigue a la instrucción LDA, en este caso $41.

Al encontrar el procesador, la línea 240, se ejecuta un salto a la subrutina ubicada en $F2B0. Esta rutina se encuentra definida ya en el Sistema Operativo del Computador y es utilizada para imprimir en la pantalla el carácter representado por el byte alojado en el Acumulador. Existe una gran cantidad de subrutinas ya definidas en el Sistema que nosotros podemos utilizar para facilitarnos la tarea. En las siguientes ediciones de TURBO news, analizaremos estas rutinas a medida que podamos aplicarlas. La ejecución de esta instrucción tiene como resultado, la impresión de una letra A en la pantalla del computador.

En la siguiente línea, el procesador se encuentra con un JMP INICIO que lo obliga a volver nuevamente a la dirección $4000 y ejecutar nuevamente todo el programa.

Con esta rutina el computador se queda indefinidamente imprimiendo letras A en la pantalla, hasta que presionemos la tecla RESET.

ASSEMBLER

Para ejecutar este programa, es necesario primero ensamblarlo. Para esto tenemos que ejecutar la instrucción ASM en el editor. Una vez tipeada la palabra ASM, presiona la tecla RETURN y en tu pantalla te aparecerá el siguiente listado:(1)

En la primera página del programa ensamblado, verás a la izquierda del listado, el programa directamente en lenguaje de máquina, y a la derecha el listado del programa en lenguaje Assembler, tal como lo hemos ingresado en el Editor. Luego de ensamblar el programa, el MAC te informará si se produjo algún error, y si no hubo ninguno podemos ejecutar el programa. Para ésto ingresa la palabra clave DDT y presiona RETURN. Al hacerlo, el MAC cambia de pantalla y deberás ingresar el comando:

* 4000, y presionar la teclas RETURN y START. Recién aquí el programa entra en ejecución, y podrás ver los resultados en tu pantalla. No olvides, que para detener el programa deberás presionar RESET.

En este artículo, hemos analizado las primeras instrucciones del Lenguaje Assembler para llegar a definir nuestra primera subrutina ejecutable. En las siguientes ediciones de tu revista TURBO news, veremos todos los conceptos que en esta lección hemos dejado de lado para poder generar nuestro primer programa en Assembler.

Nota: En caso de que tengas el Assembler editor en lugar del mac 65 debes tipear la instrucción "BUE" en reemplazo de la instrucción "DDT", a lo cual, el computador contestará "DEBUG". Luego escribe "G 4000" y presionar la tecla Return, el resto de las opciones son iguales para ambos editores.

(1)

```
PAGE 1

                  0100 ; *********************
                  0110 ; SUBRUTINA DE IMPRESION DEL
                  0120 ; CARACTER A EN LA PANTALLA
                  0130 ; *********************
                  0140 ;
                  0150 ;
0000              0160   .OPT OBJ
                  0170 ;
                  0180 ; DIRECCION DE INICIO
                  0190 ; DE LA SUBRUTINA:
                  0200 ;
0000              0210   *=$4000
4000              0220 INICIO
4000 A941         0230   LDA #$41
4002 20B0F2       0240   JSR $F2B0
4005 4C0040       0250   JMP INICIO

ASSEMBLY ERRORS: 0   28201 BYTES FREE

PAGE 2
SYMBOLS

4000 INICIO
```

CON EL ASSEMBLER PUEDE ALCANZAR NUEVAS ALTURAS EN COMPUTACION

Atari por dentro

```
10 REM ******************************
20 REM *   REVISTA TURBO NEWS (R)   *
30 REM * ARTICULO:                  *
40 REM *  ATARI POR DENTRO          *
50 REM * OBJETIVO:                  *
60 REM *  EJEMPLO DE INTERRUPCIONES *
70 REM *  EN LISTA DE DESPLIEGUE    *
80 REM *    PEDRO P. CARABALL A.    *
90 REM ******************************
100 POKE 106,128:GRAPHICS 0
110 POKE 710,0
120 FOR A=36864 TO 37206
130 READ N:POKE A,N
140 NEXT A
150 X=USR(36864)
160 LIST
170 END
1000 DATA 104,169,64,141,14,212,169,128,141,7,212,169,2,141,29,208,162,2,169,2
1010 DATA 157,8,208,169,15,157,192,2,202,16,243,169,1,141,111,2,173,48,2,133
1020 DATA 208,173,49,2,133,209,160,2,177,208,24,105,128,145,208,160,20,177,208,24
1030 DATA 105,128,145,208,169,155,141,0,2,169,144,141,1,2,162,0,138,157,244,129
1040 DATA 157,118,130,157,248,130,202,208,244,162,17,189,235,144,157,28,130,189,253,144
1050 DATA 157,156,130,189,15,145,157,28,131,189,33,145,157,98,130,189,51,145,157,226
1060 DATA 130,189,69,145,157,98,131,202,16,217,169,0,133,20,169,42,141,47,2,141
1070 DATA 0,212,169,1,205,11,212,208,251,169,192,141,14,212,96,72,165,20,141,0
1080 DATA 208,24,105,15,141,1,208,24,105,14,141,2,208,169,187,141,0,2,169,144
1090 DATA 141,1,2,104,64,0,0,72,206,186,144,206,186,144,173,186,144,141,0,208
1100 DATA 24,105,8,141,1,208,24,105,7,141,2,208,169,40,141,18,208,141,19,208
1110 DATA 141,20,208,169,155,141,0,2,169,144,141,1,2,104,64,60,66,165,129,165
1120 DATA 153,66,189,153,165,36,36,231,0,0,0,0,0,60,66,165,129,129,153,66
1130 DATA 189,153,165,36,36,231,0,0,0,0,0,60,66,165,129,153,165,66,189,153
1140 DATA 165,36,36,231,0,0,0,0,0,0,31,31,4,15,56,115,196,196,115
1150 DATA 56,15,4,31,31,0,0,3,3,19,16,63,224,14,16,36,36,16,14,224
1160 DATA 63,16,19,3,3,240,240,240,192,240,8,236,172,168,168,172,236,8,240,192
1170 DATA 240,240,240
```

LISTADO 2

```
0100 **************************
0110 * REVISTA TURBO NEWS (R) *
0120 *                        *
0130 * ARTICULO: ATARI POR    *
0140 *            DENTRO      *
0150 *                        *
0160 * OBJETIVO: MUESTRA DE   *
0170 * DOS INTERRUPCIONES EN  *
0180 * LISTA DE DESPLIEGUE.   *
0190 *                        *
0200 * ESTE PROGRAMA SOLO ES  *
0210 * INFORMATIVO YA QUE SE  *
0220 * ENCUENTRA INCORPORADO  *
0230 * EN EL PROGRAMA BASIC   *
0240 * COMO DATA.             *
0250 *                        *
0260 *  PEDRO P. CARABALL A.  *
0270 *                        *
0280 **************************
0290 ;
0300      .OPT NO LIST
0310      .OPT OBJ
```

```
0320 ;
0330      *=  $9000    ; DIRECCION DE CO
MPILACION
0340 ;
0350 ;
0360 ;
0370      PLA          ; No. ARGUMENTOS
0380      LDA #$40     ; POR AHORA NO
0390      STA $D40E    ; INTERRUPCIONES!
0400      LDA #$80     ; PM BASE
0410      STA $D407    ; EN $8000
0420      LDA #$02     ; BIT PLAYERS
0430      STA $D01D    ; CONECTADO!
0440      LDX #$02     ; PLAYER 0,1,2
0450 PORTEYCOLOR
0460      LDA #$02     ; PORTE NORMAL
0470      STA $D008,X ; PLAYER X
0480      LDA #$0F     ; COLOR BLANCO
0490      STA $02C0,X ; PLAYER X
0500      DEX          ; SIGUIENTE
0510      BPL PORTEYCOLOR ; TODOS?
0520      LDA #$01     ; PRIORIDAD
0530      STA $026F    ; PLAYERS
0540      LDA $0230    ; LISTA DE
0550      STA $D0      ; DESPLIEGUE
0560      LDA $0231    ; A $D0 PARA
0570      STA $D1      ; MODIFICAR
0580      LDY #$02     ; AL TERCER
0590      LDA ($D0),Y ; BYTE SE LE
0600      CLC          ; SUMA 128
0610      ADC #$80     ; PARA PERMITIR
0620      STA ($D0),Y ; INTERRUPCION
0630      LDY #$14     ; BYTE 21 (IDEM)
0640      LDA ($D0),Y ; EN LISTA DE
0650      CLC          ; DESPLIEGUE
0660      ADC #$80     ; PERMITE SEGUNDA
0670      STA ($D0),Y ; INTERRUPCION
0680      LDA # <DLI   ; APUNTA VECTOR
0690      STA $0200    ; INTERRUPCION
0700      LDA # >DLI   ; A RUTINA
0710      STA $0201    ; DLI
0720      LDX #$00     ; LOOP
0730      TXA          ; DE BORRADO
0740 LOOPERASE ;       AREA PLAYERS
0750      STA $81F4,X ; PLAYER 0
0760      STA $8276,X ; PLAYER 1
0770      STA $82F8,X ; PLAYER 2
0780      DEX          ; 256 BYTES
0790      BNE LOOPERASE ; EN BLANCO
0800      LDX #$11     ; FORMA PLAYERS
0810 LOOPFORMA
0820      LDA DATA1,X ; FORMA 1
0830      STA $821C,X ; PLAYER 0
0840      LDA DATA2,X ; FORMA 2
0850      STA $829C,X ; PLAYER 1
0860      LDA DATA3,X ; FORMA 3
0870      STA $831C,X ; PLAYER 2
0880      LDA DATA4,X ; FORMA 4
0890      STA $8262,X ; PLAYER 0
0900      LDA DATA5,X ; FORMA 5
0910      STA $82E2,X ; PLAYER 1
0920      LDA DATA6,X ; FORMA 6
0930      STA $8362,X ; PLAYER 2
0940      DEX
0950      BPL LOOPFORMA
0960      LDA #$00     ; RECOMIENZA
0970      STA $14      ; CONTEO RELOJ
0980      LDA #$2A     ; HABILITA
0990      STA $022F    ; PLAYERS
1000      STA $D400    ; AHORA!
1010      LDA #$01     ; ESPERA PRIMERA
1020 ESPERA ;          LINEA DE BARRIDO
1030      CMP $D40B    ; PARA PRIMERA
1040      BNE ESPERA   ; INTERRUPCION
1050      LDA #$C0     ; PERMITE
1060      STA $D40E    ; INTERRUPCION
1070      RTS          ; AL BASIC.
1080 DLI ;             PRIMERA INTERRUPC
ION
1090      PHA          ; GUARDA ACUMULAD
OR
1100      LDA $14      ; RELOJ=POSICION
1110      STA $D000    ; PLAYER 0
1120      CLC
1130      ADC #$0F
1140      STA $D001    ; PLAYER 1
1150      CLC
1160      ADC #$0E
1170      STA $D002    ; PLAYER 2
1180      LDA # <DLI1  ; APUNTA VECTOR
```

```
1190      STA $0200    ; INTERRUPCION
1200      LDA # >DLI1  ; A RUTINA
1210      STA $0201    ; DLI1
1220      PLA          ; RECUPERA ACUMUL
ADOR
1230      RTI          ; RETORNO
1240 POS0 .BYTE 0      ; POSICION SEGUND
A
1250 POS1 .BYTE 0      ; INTERUPCION
1260 DLI1 ;            SEGUNDA INTERRUPC
ION
1270      PHA          ; GUARDA ACUMULAD
OR
1280      DEC POS1     ; MOVER 2 PIXELS
1290      DEC POS1     ; A LA IZQUERDA
1300      LDA POS1     ; POSICION ACTUAL
1310      STA $D000    ; PLAYER 0
1320      CLC
1330      ADC #$08
1340      STA $D001    ; PLAYER 1
1350      CLC
1360      ADC #$07
1370      STA $D002    ; PLAYER 2
1380      LDA #$28     ; COLOR
1390      STA $D012    ; PLAYER 0
1400      STA $D013    ; PLAYER 1
1410      STA $D014    ; PLAYER 2
1420      LDA # <DLI   ; APUNTAR VECTOR
1430      STA $0200    ; A PRIMERA
1440      LDA # >DLI   ; RUTINA DE
1450      STA $0201    ; INTERRUPCION
1460      PLA          ; RECUPERA ACUMUL
ADOR
1470      RTI          ; RETORNO
1480 DATA1 ;           HOMBRE SONRIENTE
1490      .BYTE $3C,$42,$A5,$81,$A5,$99
,$42,$BD,$99,$A5,$24,$24,$E7,$00,$00,$
00,$00,$00
1500 DATA2 ;           HOMBRE SERIO
1510      .BYTE $3C,$42,$A5,$81,$81,$99
,$42,$BD,$99,$A5,$24,$24,$E7,$00,$00,$
00,$00,$00
1520 DATA3 ;           HOMBRE ENOJADO
1530      .BYTE $3C,$42,$A5,$81,$99,$A5
,$42,$BD,$99,$A5,$24,$24,$E7,$00,$00,$
00,$00,$00
1540 DATA4 ;           PUNTA AUTO
1550      .BYTE $00,$00,$1F,$1F,$04,$0F
,$38,$73,$C4,$C4,$73,$38,$0F,$04,$1F,$
1F,$00,$00
1560 DATA5 ;           CENTRO AUTO
1570      .BYTE $03,$03,$13,$10,$3F,$E0
,$0E,$10,$24,$24,$10,$0E,$E0,$3F,$10,$
13,$03,$03
1580 DATA6 ;           COLA AUTO
1590      .BYTE $F0,$F0,$F0,$C0,$F0,$08
,$EC,$AC,$A8,$A8,$AC,$EC.$08,$F0,$C0,$
F0,$F0,$F0
```

juegos

```
10 REM *****************************
20 REM *    MIS PRIMERAS LETRAS     *
22 REM *                            *
23 REM *  REVISTA TURBO NEWS (R)     *
24 REM *                            *
25 REM *                            *
26 REM * OBJETIVO:JUEGO DIDACTICO   *
27 REM *          PARA RECONOCER    *
28 REM *          PARES DE LETRAS   *
29 REM *          Y NUMEROS.        *
30 REM *                            *
31 REM *    PEDRO P. CARABALL A.    *
32 REM *****************************
100 GRAPHICS 2:POKE 710,0:POKE 708,130
:POKE 82,0
120 DL=PEEK(560)+PEEK(561)*256
130 POKE DL+13,PEEK(DL+13)+128
140 DLI=ADR("H)◆▲T .T)(■ Phe")
150 PRT=ADR("hh,Uh,T┘ZXi┘,PZYi♥,Q ♥"\1
TAH§┤)HP )♥,PfPP fQhJ┘i┘ZPi ,PZQi♥,QHC
PT◆")
160 HI=INT(DLI/256):LO=DLI-HI*256
170 POKE 512,LO:POKE 513,HI
180 FOR A=1536 TO 1543:POKE A,0:POKE A
+8,255:NEXT A:POKE A+13,255:POKE A+15,
255
182 POKE 54286,192:POKE 756,6:Z=0:K=3
184 FOR A=25 TO 16 STEP -1:X=USR(PRT,A
*8+57344):SOUND 0,15,14,10:W=A/2:GOSUB
 500:W=50:GOSUB 500:NEXT A
```

```
190 X=USR(PRT,57344):FOR A=100 TO 0 STEP -1:SOUND 0,A,12,A/10:POKE 712,A:POKE 710,A:NEXT A
191 ? "K↓ 0123456789 ABCDEFGHIJKLMNOPQRSTUVWXYZ↑"
192 ? #6;"||||||||||||||||||||"
195 POSITION 3,9:? #6;"puntos : 0"
200 L=INT(RND(0)*36):L=L+(L<10)*16+(L>9)*23:IF L=Z THEN 200
210 X=USR(PRT,L*8+57344):POKE 542,200
215 IF  NOT PEEK(644) THEN 215
220 J=PEEK(632):J=(J=11)*30+(J=7)*31
230 IF J THEN ? CHR$(J);:SOUND 0,10,12,10:W=2:GOSUB 500
240 IF PEEK(644) THEN 220
260 Z=PEEK(PEEK(660)+PEEK(661)*256+PEEK(657)+40)-128
270 IF L=Z THEN 300
280 K=K-1:SOUND 0,200,10,10:W=50:GOSUB 500:GOTO 215
300 R=P+INT(PEEK(542)/5):R=R+K:K=3:FOR A=P TO R:POSITION 12,9:? #6;A:SOUND 0,10,10,10:W=2:GOSUB 500:NEXT A:P=R
310 GOTO 200
500 POKE 540,W
510 IF PEEK(540) THEN 510
520 SOUND 0,0,0,0:RETURN
```

LISTADO 2

```
0100 ***************************
0110 * REVISTA TURBO NEWS (R) *
0120 *  MIS PRIMERAS LETRAS   *
0130 *                        *
0140 * OBJETIVO: DESPLIEGUE   *
0150 * CARACTER 16 VECES MAS  *
0160 * GRANDE EN GRAFICO DOS. *
0170 *                        *
0180 * ESTE PROGRAMA SOLO ES  *
0190 * INFORMATIVO YA QUE SE  *
0200 * ENCUENTRA INCORPORADO  *
0210 * EN EL PROGRAMA BASIC   *
0220 * (RUTINA PRT).          *
0230 *                        *
0240 *  PEDRO P. CARABALL A.  *
0250 *                        *
0260 **************************
0270      .OPT OBJ
0280      .OPT NO LIST
0290 CHAR =  $D4
0300 PANT =  $58
0310 PRIN =  $D0
0320      *=  $0600   ; PARA COMPILAR
0330      PLA         ; No. ARGUMENTOS
0340      PLA         ; ARG. 1 HI
0350      STA CHAR+1  ; POSICION CARACTER
0360      PLA         ; ARG. 1 LO
0370      STA CHAR
0380      CLC         ; BORRA ACARREO
0390      LDA PANT    ; POSICION PANTALLA
0400      ADC #$1A    ; SUMA 1 LINEA
0410      STA PRT     ; Y SEIS CARACTERES
0420      LDA PANT+1  ; BYTE ALTO ORDEN
0430      ADC #$00    ; SUMA ACARREO
0440      STA PRT+1
0450      LDY #$00    ; PRIMER BYTE
0460 LOOP
0470      LDX #$07    ; OCHO BITS
0480      LDA (CHAR),Y ; LEE BYTE
0490 LOOP1
0500      ASL A       ; REVISA BIT
0510      PHA         ; GUARDA ACUMULADOR
0520      BCC VACIO   ; BIT = 0?
0530      LDA #$01    ; BIT=1!
0540      BNE LLENO   ; SALTAR VACIO
0550 VACIO
0560      LDA #$00    ; BIT=0!
0570 LLENO
0580      STA (PRT),Y ; IMPRIME
0590      INC PRT     ; AUMENTA POS.
0600      BNE NOCARRY ; DE IMPRESION
0610      INC PRT+1   ; EN 1 CARACTER
0620 NOCARRY
0630      PLA         ; RECUPERA ACUMULADOR
0640      DEX         ; 1 BIT LISTO!
0650      BPL LOOP1   ; 8 BITS?
```

```
0660      CLC          ; AUMENTA
0670      LDA PRT      ; POCISION DE
0680      ADC #$0B     ; IMPRESION
0690      STA PRT      ; A LA PROXIMA
0700      LDA PRT+1    ; LINEA
0710      ADC #$00
0720      STA PRT+1
0730      INY          ; 1 BYTE LISTO!
0740      CPY #$08     ; 8 BYTES LISTOS?
0750      BNE LOOP     ; NO, CONTINUE!
0760      RTS          ; RETORNO AL BASI
C
0770 ;
0780 ;
0790 ;
0800 ;
0810 ;
0820 ;  RUTINA DE INTERRUPCION
0830 ;  DE LISTA DE DESPLIEGE
0840 ;
0850 ;  (EN PROGRAMA BASIC DLI)
0860 ;
0870      PHA          ; GUARDA ACUMULAD
OR
0880      LDA #$E0     ; BASE CARACTERES
0890      STA $D40A    ; COMIENZO LINEA
0900      STA $D409    ; CARECTERES
0910      LDA #$28     ; COLOR NUMEROS
0920      STA $D016    ; INMEDIATAMENTE!
0930      PLA          ; RECUPERA ACUMUL
ADOR
0940      RTI          ; FIN INTERRUPCIO
N
```

Rutina 1

```
0 O=ADR("hh▛▘h▜ ▯R▄▖▯U▟▗▯N▄▝▯x▟▙▯|
▜▖LV▮"):U=USR(ADR("hh▛▘h▜ |♦"),0):POKE
O+23,155:POKE 9,2:U=USR(ADR("hl" ♥♦"))
10 REM ****************************
20 REM *  REVISTA TURBO NEWS (R)   *
30 REM * RUTINA PARA RECOMENZAR LA *
40 REM * EJECUCION DEL PROGRAMA    *
50 REM * BASIC, SI ES PRESIONADA   *
60 REM * LA TECLA RESET.           *
70 REM *    PEDRO P. CARABALL A.   *
80 REM ****************************
100 REM  PROGRAMA DE EJEMPLO
110 GRAPHICS 18
120 POKE 580,0
130 POSITION 1,2
140 ? #6;"ME RIO DE LA TECLA"
150 POSITION 8,6
160 ? #6;"RESET"
170 POKE 710,PEEK(20):GOTO 170
```

Rutina 2

```
0 REM ****************************
1 REM *  REVISTA TURBO NEWS (R)   *
2 REM * RUTINA PARA LEER  DESDE   *
3 REM * UN PERIFERICO A VELOCIDAD *
4 REM * DE LENGUAJE DE MAQUINA    *
5 REM * DESDE EL BASIC.           *
6 REM *     PEDRO P. CARABALL A.  *
7 REM ****************************
10 CIOV=ADR("h▯H▯▚▯0▖hh8▯▯♦hh▚▚▚▚▮▚▘▯
hh▗▘▯B▘h▯E▘h▯D▘h▯I▘h▯H▘ V▯▚▘▯H▘▜▯I▘▜
▯♦")
20 REM X=USR(CIOV,CANAL,OPERACION[4-8]
,DIRECCION,CANTIDAD):X=RESULTADO (EN B
YTES)
30 POKE 710,0:N=1
40 CANT=FRE(0)-2000:DIM A$(CANT)
50 DIR=ADR(A$)
60 ? "K} INGRESE CASSETTE A COPIAR"
70 ? "}PRESIONE  PLAY  Y  RETURN "
80 OPEN #1,4,128,"C:"
90 X=USR(CIOV,1,4,DIR,CANT):CANT=X
100 IF PEEK(63) THEN 120
110 ? :? "▯}MEMORIA INSUFICIENTE":END
120 CLOSE #1
130 ? "K   INGRESE CASSETTE DESTINO No
.";N:N=N+1
140 ? " PRESIONE  PLAY RECORD  Y RET
URN"
150 OPEN #1,8,128,"C:"
160 X=USR(CIOV,1,8,DIR,CANT)
170 CLOSE #1
180 GOTO 130
```

LISTADO 2

```
0100 ************************
0110 * REVISTA TURBO NEWS (R) *
0120 *                      *
0130 *  OPERACION CIO-BASIC   *
0140 *                      *
0150 * ESTE PROGRAMA SOLO ES  *
0160 * INFORMATIVO YA QUE SE  *
0170 * ENCUENTRA INCORPORADO  *
0180 * EN EL PROGRAMA BASIC   *
0190 * RUTINA CIOV.           *
0200 *                      *
0210 *  PEDRO P. CARABALL A.  *
0220 ************************
0230      .OPT NO LIST
0240      .OPT OBJ
0250 ;
0260 ; desde basic
0270 ; x=usr(n,a,b,c,d)
0280 ; a=canal
0290 ; b=operación (4=read 8=write)
0300 ; c=dirección
0310 ; d=cantidad
0320 ;
0330      *=  $0600
0340      PLA          ; No. ARGS.
0350      CMP #4       ; CUATRO?
0360      BEQ CORRECTO ; SI!
0370      TAX          ; NO!
0380 ERROR
0390      DEX          ; DESCARTA
0400      BMI BASIC    ; TODOS LOS
0410      PLA          ; ARGUMENTOS
0420      PLA          ; Y RETORNA
0430      SEC          ; AL BASIC
0440      BCS ERROR    ; SIN EJECUTAR
0450 BASIC
0460      RTS          ; NADA!
0470 CORRECTO
0480      PLA          ; CANAL HI = #$00
0490      PLA          ; CANAL
0500      ASL A        ; *2
0510      ASL A        ; *4
0520      ASL A        ; *8
0530      ASL A        ; *16
0540      STA $0307    ; AUXILIAR
0550      TAX          ; REGISTRO DE CAN
AL
0560      PLA          ; #$00
0570      PLA          ; TRADUCE
0580      ORA #$03     ; OPERACION
0590      STA $0342,X
0600      PLA          ; BUFFER HI
0610      STA $0345,X
0620      PLA          ; BUFFER LO
0630      STA $0344,X
0640      PLA          ; CANTIDAD HI
0650      STA $0349,X
0660      PLA          ; CANTIDAD LO
0670      STA $0348,X
0680      JSR $E456    ; EJECUTE CIOV
0690      LDX $0307    ; RECUPERA CANAL
0700      LDA $0348,X ; RESPUESTA AL
0710      STA $D4      ; BASIC,
0720      LDA $0349,X ; BYTES
0730      STA $D5      ; TRASPASADOS.
0740      RTS          ; AL BASIC!
```

Rutina 3

```
0 REM ****************************
1 REM *  REVISTA TURBO NEWS (R)   *
2 REM * RUTINA PARA ESCRIBIR O    *
3 REM * LEER UN SECTOR DEL DISCO  *
4 REM * DESDE EL BASIC.           *
5 REM *     PEDRO P. CARABALL A.  *
6 REM ****************************
10 SECTOR=ADR("hh█┘h█◢┘h█┌┘h█├█♥█◢┘▯
R▯ █@█┘┘h█┘h█┤┘h▯h┴█├┘i0█♥┘ S▯█♦")
20 REM X=USR(SECTOR,NUMSECTOR,COMANDO[
R,P,W],BUFFER,DRIVE):X=STATUS
30 POKE 710,0:N=1
40 REM  PROGRAMA DE EJEMPLO 
50 POKE 710,0:DIM A$(128)
60 DIR=ADR(A$):OPEN #1,4,0,"K:"
70 ? "KSECTOR A LEER ";:INPUT SL
80 ? "VERIFICO ESCRITURA (s/n)?";
90 GET #1,K:IF K<>ASC("S") AND K<>ASC(
```

```
"N") THEN 90
100 ?.CHR$(K)
110 IF K=ASC("S") THEN K=ASC("W")
120 IF K=ASC("N") THEN K=ASC("P")
130 ? "SECTOR A ESCRIBIR ";:INPUT SE
140 ? :? "INGRESE DISCO A LEER,":? "PR
ESIONE UNA TECLA":GET #1,W
150 X=USR(SECTOR,SL,ASC("R"),DIR,1)
160 IF X<>1 THEN ? "ERROR ";X;" EN LE
CTURA":END
170 ? :? "INGRESE DISCO A ESCRIBIR,":?
 "PRESIONE UNA TECLA":GET #1,W
180 X=USR(SECTOR,SE,K,DIR,1)
190 IF X<>1 THEN ? "ERROR ";X;" EN ES
CRITURA":END
200 RUN
```

LISTADO 2

```
0100 ************************
0110 * REVISTA TURBO NEWS (R) *
0120 *                        *
0130 *  LECTURA SECTOR-BASIC  *
0140 *                        *
0150 * ESTE PROGRAMA SOLO ES  *
0160 * INFORMATIVO YA QUE SE  *
0170 * ENCUENTRA INCORPORADO  *
0180 * EN EL PROGRAMA BASIC   *
0190 * RUTINA SECTOR.         *
0200 *                        *
0210 *  PEDRO P. CARABALL A.  *
0220 ************************
0230      .OPT OBJ
0240      .OPT NO LIST
0250 ;
0260 ; desde basic
0270 ; x=usr(n,a,b,c,d)
0280 ; a=numero de sector
0290 ; b=comando (r=leer,p=escribir,w=
escribir y verificar)
0300 ; c=direccion de memoria
0310 ; d=numero de drive
0320      ;
0330      *=  $0600
0340 ;
0350      PLA          ; NUMERO ARGS.
0360      PLA          ; SECTOR HI
0370      STA $030B
0380      PLA          ; SECTOR LO
0390      STA $030A
0400      PLA          ; #$00
0410      STA $0309
0420      PLA          ; "R","W" o "P"
0430      STA $0302
0440      LDX #$80     ; WRITE MODE
0450      STX $0308    ; BYTES * SECTOR
0460      CMP #'R      ; READ ?
0470      BNE NOREAD
0480      LDX #$40     ; READ MODE
0490 NOREAD
0500      STX $0303    ; OPERACION
0510      PLA          ; BUFFER HI
0520      STA $0305
0530      PLA          ; BUFFER LO
0540      STA $0304
0550      PLA          ; #$00
0560      STA $D5      ; STATUS HI=$00
0570      PLA          ; NUMERO DRIVE
0580      CLC
0590      STA $0301    ; DRIVE
0600      ADC #$30
0610      STA $0300    ; DEVICE
0620 LOOPREAD
0630      JSR $E453    ; EJECUTE CIO
0640      STY $D4      ; STATUS A BASIC
0650      RTS          ; VOILA!
```

Desarrollando Hardware

```
10 REM ****************************
20 REM *   REVISTA TURBO NEWS (R)   *
30 REM * ARTICULO:                  *
40 REM *  DESARROLLANDO HARDWARE    *
50 REM * OBJETIVO:                  *
60 REM *  EJEMPLO DE UTILIZACION DE *
70 REM *  DISPLAY NUMERICO.         *
80 REM *    PEDRO P. CARABALL A.    *
90 REM ****************************
100 POKE 54018,48:POKE 54016,255:POKE
54018,60
```

```
110 GRAPHICS 0:POKE 710,0:POKE 752,1
120 ? "    1 .- RELOJ"
130 ? "    2 .- CARACTER"
140 ? "    3 .- CONTADOR"
170 OPEN #1,4,0,"K:"
180 GET #1,N:N=N-48:IF N<1 OR N>3 THEN 180
190 ON N GOTO 200,300,400
200 REM  CONTADOR DE TIEMPO 
210 ? "↖↓↓▸      CONTANDO SEGUNDOS"
220 POKE 19,0:POKE 20,0
230 N=INT((PEEK(20)+PEEK(19)*256)/60):IF N>59 THEN 220
240 GOSUB 1000:GOTO 230
300 REM  VALOR DE TECLA PRESIONADA 
310 ? "↖↓↓▸    VAROR ASCII DE TECLA"
320 GET #1,N:IF N>99 THEN 320
330 GOSUB 1000:GOTO 320
400 ? "↖↓↓▸TIEMPO TOTAL DESEADO ";:INPUT T
410 W=INT(0.6*T)
420 ? "↓↓ CONTANDO..."
430 FOR X=99 TO 0 STEP -1:POKE 540,W:N=X:GOSUB 1000
440 IF PEEK(540) THEN 440
450 NEXT X:RUN
1000 N2=INT(N/10)*16:N1=N-INT(N/10)*10:N=N1+N2:POKE 54016,N:RETURN
```



Turbo Software
LINEA LIGHT GUN PARA ATARI
Turbo Software
LINEA EDUCACIONAL PARA ATARI
Turbo Software
ATARI
La nueva generación de Software para Computadores Atari

En pocos segundos gane mucho tiempo

NUEVA

DISKETTERA ATARI XFF-551

La diskettera Atari XFF-551, impone una nueva velocidad de trabajo al computador ATARI XL/XE. Sólo unos pocos segundos bastan, para que la unidad de diskettes magnéticos de 5¼ pulgadas XFF cargue completamente un programa en la memoria de su computador. Sin demoras, sin esperas, sin errores. Para un veloz acceso a la información que necesita, incorpórela rápidamente a su computador y pase adelante a toda prisa.



ATARI®
COMPUTADORES



Ocho de cada diez computadores para el hogar, son ATARI